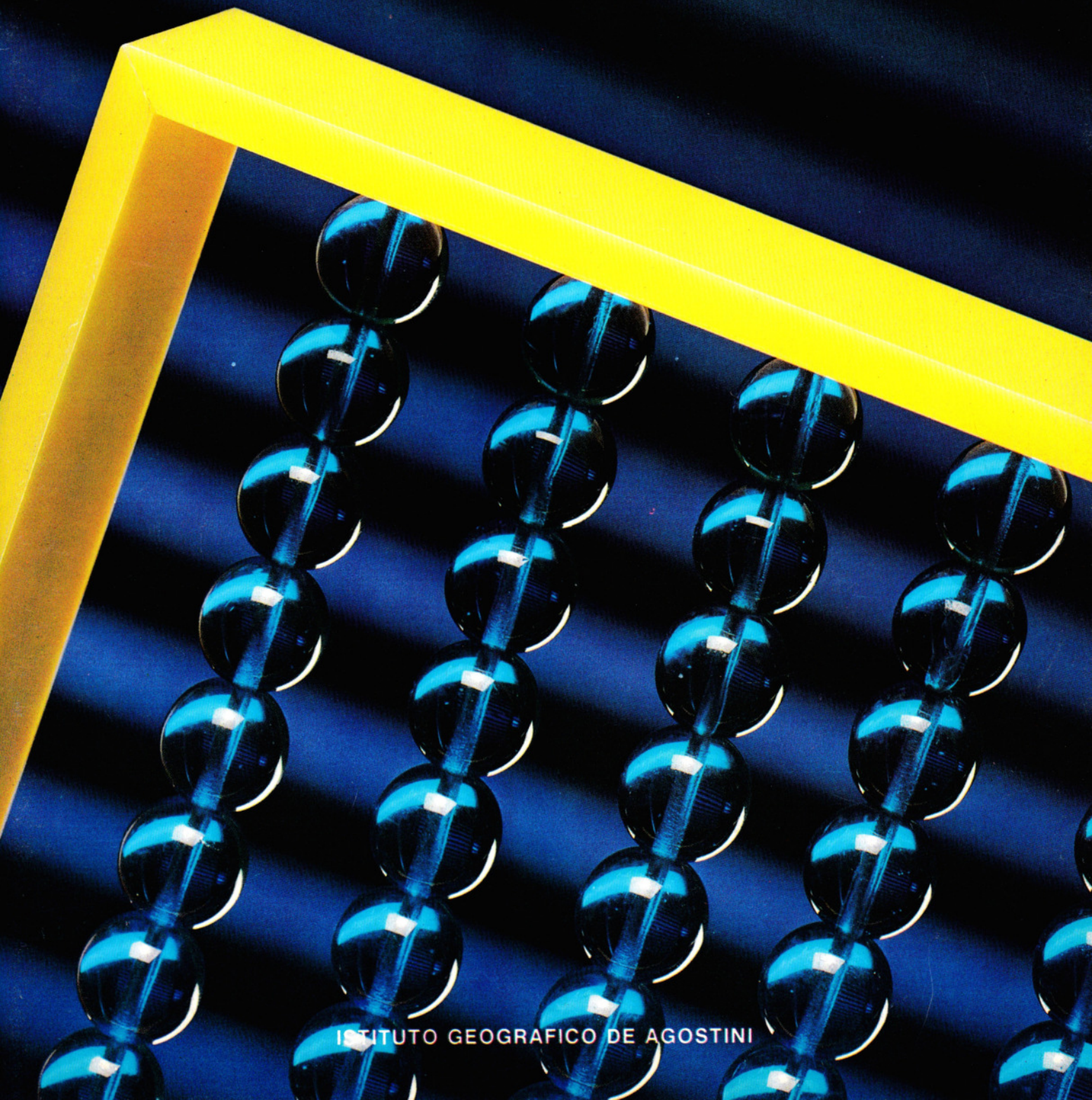
INPUT
4
L. 2800
CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER
ISTITUTO GEOGRAFICO DE AGOSTINI

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

*Direttori:* Achille Boroli - Adolfo Boroli

*Direzione editoriale:* Mario Nilo; *settore fascicoli:* Jason Vella

*Redazione dell'edizione italiana a cura della:*
Logical Studio Communication
*Traduzione dall'inglese a cura di:* Daniel Quinn

*Coordinamento grafico:* Otello Geddo

*Coordinamento fotografico* a cura del Centro Iconografico dell'Istituto Geografico De Agostini

*Direzione:* Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5

*Redazione:* Milano (20149), via Mosè Bianchi 6 - tel. (02) 4694451

*Programma di abbonamento.* Condizioni di abbonamento all'intera opera in 52 fascicoli, completa di copertine e di risguardi per la confezione dei 6 volumi dell'opera:
*a)* in un unico versamento anticipato di L. 180 000 in Italia, L. 225 000 all'estero;
*b)* in 4 versamenti trimestrali consecutivi e anticipati di L. 45 250 ciascuno.
La forma di abbonamento *b* è ammessa soltanto in Italia.
Agli abbonati all'intera opera sono riservato in dono "2 cassette di videogiochi" oppure, in alternativa, "5 cassette da registrare" (Aut. Min. conc.).
I versamenti possono essere effettuati a mezzo assegno bancario oppure sul c/c postale n. 111286 intestato all'Istituto Geografico De Agostini - Novara.
*Amministrazione, abbonamenti e servizio arretrati:* Istituto Geografico De Agostini - Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5.
Copertine e risguardi per i volumi dell'opera saranno messi in vendita a L. 6000 la copia (L. 7500 all'estero).
Le copie arretrate saranno disponibili per un anno dal completamento dell'opera e potranno essere prenotate nelle edicole o direttamente presso l'Editore. Per i fascicoli arretrati, trascorse 12 settimane dalla loro pubblicazione, è applicato un sovrapprezzo di L. 400 sul prezzo di copertina in vigore al momento dell'evasione dell'ordine. Spedizione contro rimessa di pagamento anticipato; non vengono effettuate spedizioni contrassegno.
L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.


Registrato presso il Tribunale di Novara n. 11 in data 19-5-1984.
*Direttore responsabile:* Emilio Bucciotti

Spedizione in abbonamento postale Gruppo II/70 (Autorizzazione della Direzione provinciale delle PP.TT. di Novara).
Distribuzione A. & G. Marco - Milano, via Fortezza 27 - tel. (02) 2526.
Pubblicazione a fascicoli settimanali. Esce il martedì.
Stampato in Italia - I.G.D.A. Officine Grafiche, Novara - 138411.

*Referenze dei disegni e delle fotografie:*
Copertina: Roger Payling. Pagg. 97, 98, 100, 102 Andy Leslie. Pagg. 104, 106, 108, 109 David Lloyd. Pagg. 110, 112 Roger Payling. Pag. 114 Archivio IGDA. Pag. 115 Roger Payling. Pag. 116 Chris Lyon. Pagg. 124, 126 Peter Richardson. Effetti al computer della J.D. Audio Visual.

Pubblicazione a fascicoli settimanali
edita dall'Istituto Geografico De Agostini

**volume I - fascicolo 4**



## AVVISO AI LETTORI

Per ovviare ad eventuali problemi di caricamento del 'PROGRAMMA CHAMP' relativi al Commodore 64 e allo Spectrum Sinclair, riteniamo utile segnalare alcuni accorgimenti.

Per il **Commodore 64** le soluzioni sono due:
— portare la cassetta al giro 70 e digitare SHIFT + RUN/STOP: il caricamento sarà automatico
— digitare LOAD "CHAMP" e attendere il caricamento, quindi digitare RUN e attendere la visualizzazione dell'intestazione sul monitor.

La procedura per lo **Spectrum Sinclair** è la seguente: mettere il registratore sul volume e sui toni alti, far girare la cassetta e bloccare il registratore a circa 140 giri, digitare LOAD "CHAMP" e schiacciare ENTER. Accendere il registratore e schiacciare PLAY per far caricare la prima parte. Terminata l'operazione di caricamento della prima parte, il registratore si blocca e si spegne; a questo punto digitare RUN e contemporaneamente schiacciare PLAY. Il registratore caricherà il programma completo e poi funzionerà da solo.

# ROUTINE CONTAPUNTI E CONTATEMPO

- UN CAMPO MINATO
- MANTENERE I PUNTEGGI
- IL COMPUTER TEMPORIZZATORE
- CHI SPARA PER PRIMO

**Un gioco è molto più divertente se si sa di aver solo due secondi per tornare alla base o che mancano solo dieci punti per battere un nuovo record. Tutti i giochi tipo 'arcade', possono essere più avvincenti, dotandoli di un conteggio dei punti o del tempo: bastano poche e semplici istruzioni.**

In quasi tutti i giochi al computer, occorre un qualche meccanismo per controllare il punteggio o il tempo (o ambedue). Senza di esso, infatti, non potremmo giudicare quanto si è bravi nel gioco o se stiamo migliorando rispetto ai nostri amici.

Un amico, seduto al nostro fianco, potrebbe aiutarci a tenere il punteggio, ma c'è più soddisfazione nel far sì che sia il computer a segnare i punti. E poche linee di programma in più convinceranno la macchina a ricordarsi anche dei punteggi massimi raggiunti in più partite.

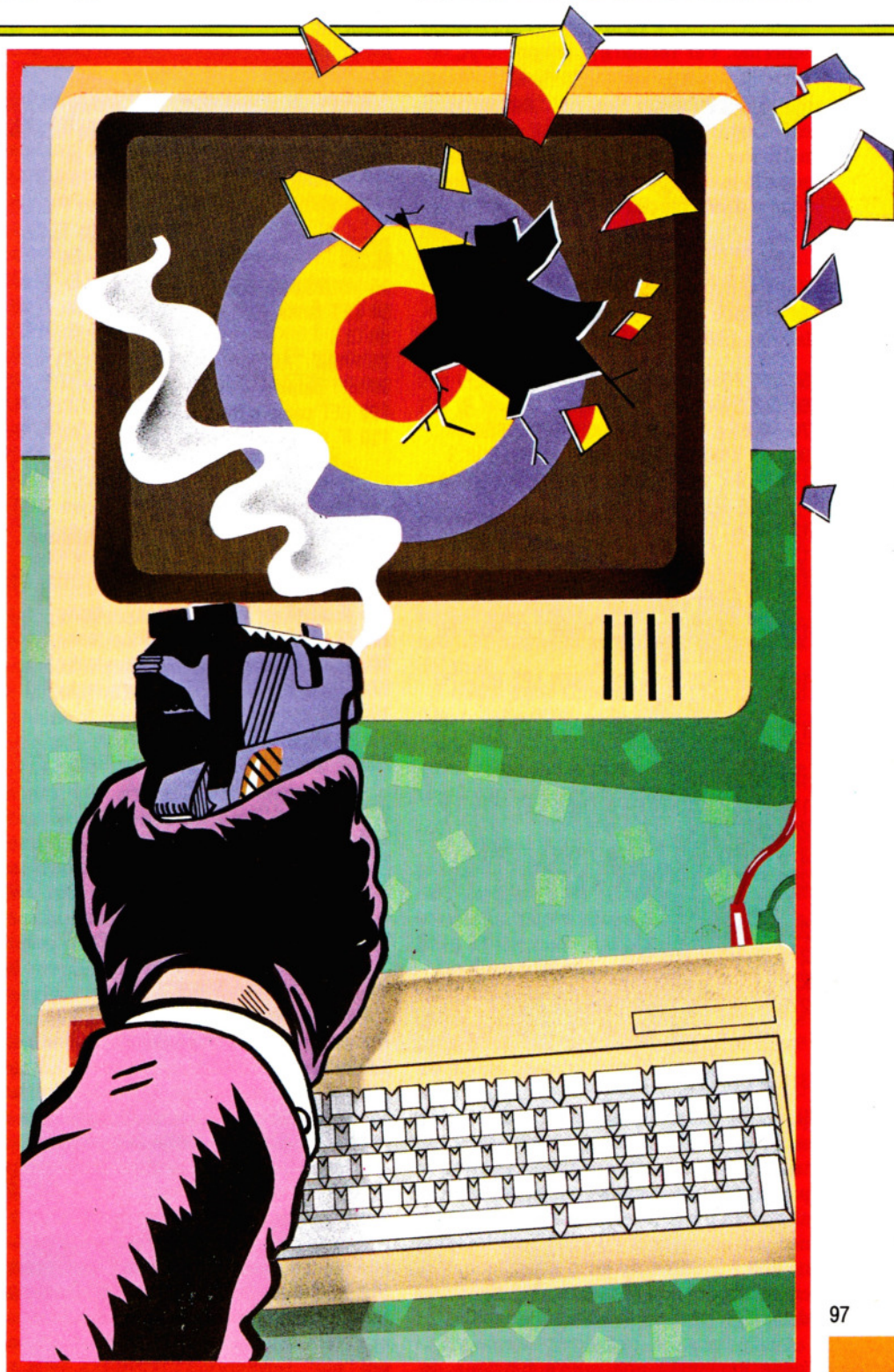
Analogamente, non occorre un cronometro per misurare il tempo. Come si è già visto parlando dei labirinti (pagine 68-74), tutte le macchine sono dotate di un timer interno, che può essere usato in molti modi per migliorare i programmi di gioco.

## UN CAMPO MINATO

Per illustrare come realizzare in pratica le routine per il punteggio e per il tempo, ecco un gioco, adatto a tutte le macchine (meno lo ZX81 e il VIC 2Ø). Ad esso aggiungeremo le due routine, talmente semplici che potremo sfruttarle comodamente anche in altri programmi.

Nel gioco, che si chiama "Campo minato", siamo al comando di un carro armato e la nostra missione è di salvare un gruppo di paracadutisti che stanno precipitando su un campo minato. Ovunque si sposti, il carro armato rischia di far saltare una mina, lasciata a caso del computer. Come in un vero campo minato, le mine sono invisibili, perciò ... prudenza!

Il carro armato (per adesso soltanto un segno #, finché non si è imparato a combinare grafica e movimento in BASIC) si controlla con i soliti tasti usati in prece-

denza: [Z] a sinistra, [X] a destra, [P] in alto e [L] in basso. Il nucleo del programma è la stessa routine di spostamento sullo schermo, vista alle pagine 54-59.

Una volta trascritto e lanciato il programma, si noterà la sua parziale incompletezza: raccattato un paracadutista, infatti, non accade nient'altro e il carro armato continua a girare senza meta sullo schermo. Per fermare il programma, occorre premere i tasti [BREAK] o [ESCAPE] o [RUN/STOP] (a meno che, nel frattempo, il carro armato non abbia scovato... una mina!). Per completarlo, aggiungeremo in seguito le due routine contapunti e contatempo descritte più avanti.

Sul Tandy, usare 247 al posto di 223 nelle linee 14Ø e 15Ø; sostituire 239 con 251 nella linea 16Ø e, nella linea 17Ø, 247 con 253.

```
50 LET PO=210
60 CLS
70 PRINT@ 256, STRING$ (32,"–")
90 LET X=RND(256)–1
110 IF X<> PO THEN PRINT@ X, "O";
    ELSE GOTO 90
120 PRINT@ PO,"#";
130 LET LP=PO
140 IF PEEK(340)=223 THEN LET PO=PO
    –1:GOTO 190
150 IF PEEK(338)=223 THEN LET PO=PO
    +1:GOTO 210
160 IF PEEK(338)=239 THEN LET PO=PO
    –32:GOTO 220
170 IF PEEK(342)=247 THEN LET PO=PO
    +32:GOTO 220
180 GOTO 140
190 IF (LP AND 31)=0 THEN LET PO=LP
200 GOTO 220
210 IF (PO AND 31)=0 THEN LET PO=LP
220 IF PO>255 OR PO<0 THEN LET PO
    =LP:GOTO 140
230 PRINT@ LP,"□";
240 PRINT@ PO,"#";
250 LET M=RND(256)–1
270 IF M=PO THEN PRINT@ PO,"□":
    PRINT@ 130,"BOOM!!–È SCOPPIATA
    UNA MINA!":STOP
310 GOTO 130
```

Il carro armato viene manovrato, sullo schermo, dalle linee da 14Ø a 17Ø, che controllano la pressione dei tasti (vedere a pagina 59). Onde limitare gli spostamenti alla metà superiore dello schermo, la linea 22Ø contiene IF PO<255... (la locazione 255 è l'ultima di detta zona), quindi il carro armato non scenderà mai al di sotto della riga tratteggiata dalla linea 7Ø.

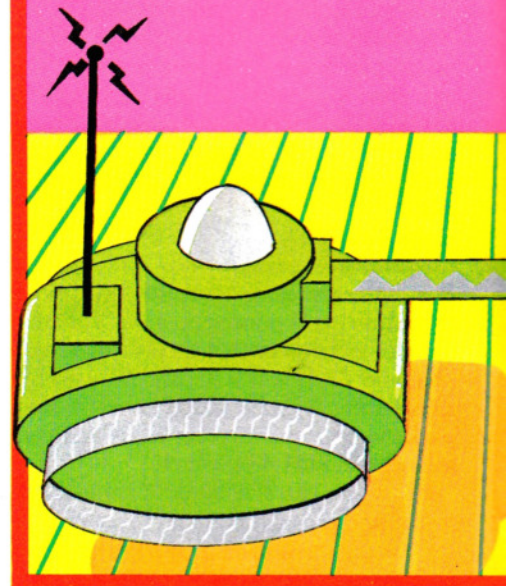
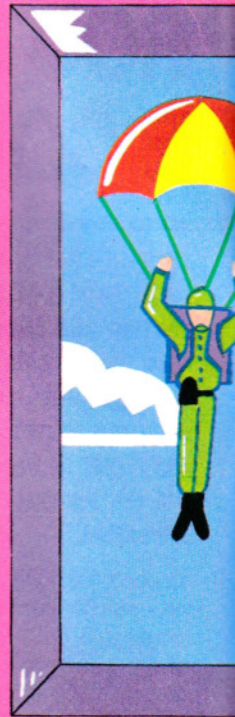
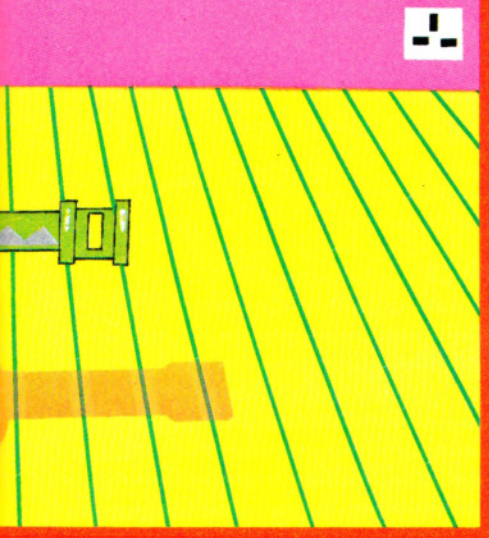
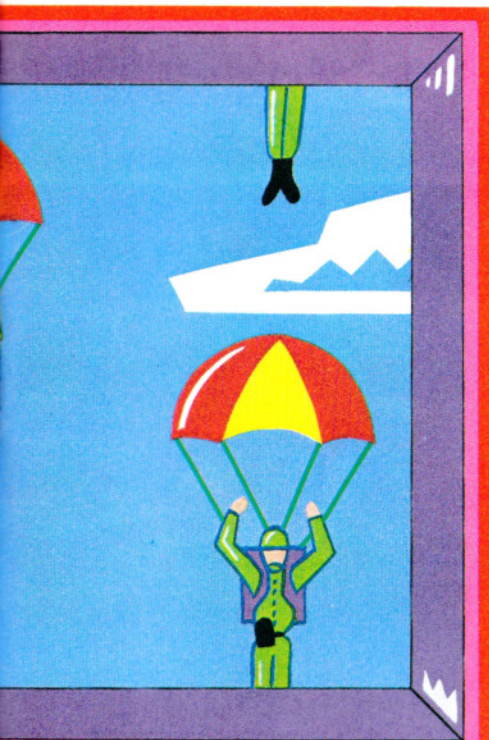
Nella linea 7Ø, per disegnare la riga, viene impiegata una nuova funzione, la STRING$. STRING$(32,"–") dice semplicemente al computer di disegnare una sequenza di 32 trattini. Se volessimo ottenere 1Ø punti interrogativi dovremmo usare STRING$(1Ø,"?") e via dicendo.

Il luogo dove atterrano i paracadutisti è scelto, usando un numero casuale, nella linea 9Ø. La linea 11Ø visualizza il paracadute, purché la posizione non sia già occupata dal carro armato, nel qual caso la linea 9Ø sceglie un un altro valore.

Infine, le posizioni delle mine vengono stabilite dalla linea 25Ø. La linea 27Ø controlla che il carro armato non si trovi sopra una mina: se ciò accade, compare un messaggio appropriato e il gioco termina.

```
15 VDU23;8202;0;0;0;
50 LET tankx=20 : LET tanky=10
60 CLS
70 PRINT TAB(0,13) STRING$ (40,"–")
90 LET parax=RND(30)+4
100 LET paray=RND(12)
110 IF (parax<>tankx) AND (paray<
    >tanky) THEN PRINT TAB(parax,paray)
    "O";ELSE GOTO 90
120 PRINT TAB(tankx,tanky) "#";
130 LET lasttankx=tankx : LET lasttantky
    =tanky
140 KEY$=GET$
145 IF KEY$="P" THEN tanky=tanky–1
150 IF KEY$="L" THEN tanky=tanky+1
160 IF KEY$="Z" THEN tankx=tankx–1
170 IF KEY$="X" THEN tankx=tankx+1
190 IF tankx<5 OR tankx>
    34 THEN LET tankx=lasttankx
200 IF tanky<1 OR tanky>12 THEN LET
    tanky=lasttanky
230 PRINT TAB(lasttankx,lasttanky) "□"
240 PRINT TAB(tankx,tanky) "#"
250 LET minex=RND(30)+4
260 LET miney=RND(12)
270 IF (minex=tankx) AND (miney=
tanky) THEN PRINT TAB(minex,miney)
"□": PRINT TAB(7,7)"BOOM!!–
È SCOPPIATA UNA MINA!":END
310 GOTO130
```

Nel programma per gli Acorn, le linee da 145 a 17Ø controllano quale tasto è stato premuto dal giocatore. Una spiegazione completa del procedimento è stata data a pagina 55. Le linee 19Ø e 2ØØ confinano i movimenti del carro armato nella parte superiore dello schermo, delimitata da una riga tratteggiata dalla linea 7Ø.

Questa riga viene tracciata usando una nuova funzione: STRING$. Scrivendo STRING$(4Ø,"–") comunichiamo al computer di visualizzare una stringa di 4Ø trattini. Volendo ottenere una serie di 15 segni "più", scriveremo STRING$(15,"+").

Il luogo d'atterraggio dei paracadutisti è scelto dalle linee 9Ø e 1ØØ. Questo viene confrontato con la posizione del carro armato nella linea 11Ø. Se le due posizioni differiscono, allora il paracadutista viene regolarmente visualizzato, altrimenti è scelto un nuovo luogo di atterraggio, tornando alla linea 9Ø.

In modo simile, le linee 250 e 260 scelgono dove collocare le mine. La linea 270 controlla se il carro armto e la mina occupano la medesima posizione e, se questo è il caso, il carro armato viene "cancellato", visualizzando al suo posto un carattere "spazio": la comparsa di un messaggio appropriato termina il gioco.

Infine, per ripulire lo schermo, alla linea 15 è impiegato un comando VDU, che spegne il cursore.

```
50 LET tx=16: LET ty=5
60 CLS
70 PRINT AT 11,0;"--------------------------------"
90 LET px=INT (RND*30)+1
100 LET py=INT (RND*10)
110 IF px=tx AND py=ty THEN GOTO 90
120 PRINT AT py,px;"O";AT ty,tx;"#"
130 LET txx=tx: LET tyy=ty
140 LET a$=INKEY$
145 IF a$="p" THEN LET ty=ty-1
150 IF a$="l" THEN LET ty=ty+1
160 IF a$="z" THEN LET tx=tx-1
170 IF a$="x" THEN LET tx=tx+1
190 IF ty<0 OR ty>10 THEN LET ty=tyy
200 IF tx<1 OR tx>30 THEN LET tx=txx
230 PRINT AT tyy,txx;"□"
240 PRINT AT ty,tx;"#"
250 LET mx=INT (RND*30)+1
260 LET my=INT (RND*10)
270 IF mx=tx AND my=ty THEN PRINT
    AT my,mx;"□":PRINT AT 8,3; "BOOM!!-È
    SCOPPIATA UNA MINA!": STOP
310 GOTO 130
```

Nel programma per lo Spectrum, lo schermo viene suddiviso in due parti da una riga tracciata dalla linea 70 (una serie di 32 trattini). Le linee da 145 a 170 controllano il movimento del carro armato, come spiegato ampiamente a pagina 57.

Le linee 190 e 200 confinano il movimento del carro all'area superiore dello schermo, evitando la sua fuoriuscita dal campo visivo. Il luogo d'atterraggio dei paracadutisti è scelto, a caso, dalle linee 90 e 100. La linea 110 controlla se un paracadutista e il carro armato occupano la stessa posizione. Se ciò avviene, la linea 90 sceglie una nuova posizione d'atterraggio.

Analogamente, le linee 250 e 260 scelgono dove collocare le mine e la linea 270 verifica se carro e mina occupano la medesima posizione. In tal caso, il carro viene cancellato (con uno spazio) e appare un messaggio che termina il gioco.

```
15 Z$="■■■■■■■■■■■■":POKE 650,128
50 LET PO=205
60 PRINT "■";Z$;"----------------------------------------"
70 POKE 53280,1:POKE 53281,3
85 LET L=5
90 LET X=INT(RND(1)*479)+1
110 IF X=PO THEN GOTO 90
115 POKE 1024+X,87
120 POKE 1024+PO,35
130 LET LP=PO: LET LL=L:GET A$
140 IF A$="Z" THEN LET PO=PO-1: L
    =L-1: GOTO 190
150 IF A$="X" THEN LET PO=PO+1: L
    =L+1: GOTO 210
160 IF A$="P" THEN LET PO=PO-40:
    GOTO 220
170 IF A$="L" THEN LET PO=PO+40:
    GOTO 220
180 GOTO 130
190 IF L<0 THEN LET PO=LP:LET L=LL
200 GOTO 220
210 IF L>39 THEN LET PO=LP:LET L=LL
220 IF 1024+PO<1024 OR 1024+PO>
    1024+479 THEN LET PO=LP:GOTO130
230 POKE 1024+LP,32
240 POKE 1024+PO,35
250 LET M=INT(RND(1)*479)+1
270 IF M=PO THEN POKE 1024+PO,32:
    PRINT Z$;"■"BOOM!!-
    È SCOPPIATA UNA MINA!": STOP
310 GOTO 130
```

Nel programma del Commodore lo schermo è suddiviso in due metà da una riga tracciata dalla linea 60. Alla variabile Z$ è assegnata una stringa di caratteri di controllo (linea 15). La POKE, sempre alla linea 15, attiva la ripetizione automatica dei tasti.

I colori dello schermo e del bordo sono fissati dalle POKE nella linea 70.

La linea 85 inizializza L, che viene in seguito usata (linee 190 e 210) per evitare che il carro esca dal campo visivo.

Il luogo d'atterraggio del paracadutista viene scelto, a caso, dalla linea 90. Tale posizione viene confrontata con quella del carro armato e, se coincidono, la linea 90 sceglie un nuovo luogo d'atterraggio. La linea 115 visualizza il paracadutista.

La linea 220 evita che il carro superi i margini dello schermo o la linea tratteggiata. La linea 230 "pulisce" la posizione precedente del carro armato e la linea 240 visualizza quest'ultimo in quella nuova.

Infine, la linea 250 seleziona una posizione a caso per la mina. La linea 270 la confronta con la posizione del carro e, se coincidono, il carro armato viene cancellato e, alla comparsa di un messaggio appropriato, il gioco finisce.

## PUNTEGGIO

In genere, nei giochi tipo 'arcade', il punteggio aumenta quando le posizioni sullo schermo di due oggetti coincidono. Gli og-

getti possono essere un missile e un obiettivo, un Pacman e una pillola energetica, un carro armato e un paracadutista, un cavallo e un traguardo o qualsiasi cosa serva al gioco.

Così, si aggiungano queste linee al programma per vedere come funziona in pratica il meccansimo contapunti:

```
40 LET S=0
280 IF X=PO THEN LET S=S+1:GOTO 90
330 PRINT@295,S;"PARACADUTISTI";
```

```
40 LET S=0
280 IF (parax=tankx) AND (paray=tanky)
   THEN LET S=S+1:GOTO 90
330 PRINT TAB(12,15);S;"□PARACADUTISTI"
```

```
40 LET s=0
280 IF px=tx AND py=ty THEN LET s=s
    +1:GOTO 90
330 PRINT AT 14,8;s;"□PARACADUTISTI"
```

```
40 LET S=0
280 IF X=PO THEN LET S=S+1:GOTO 90
330 POKE53280,254:POKE53281,246:
   PRINT "▣□□□□□□□□";S;
   "PARACADUTISTI"
```

Ora, sostituire alla linea 270 STOP o END con GOTO 330

```
270 IF M=PO THEN PRINT@ PO,"□":
   PRINT@ 130,"BOOM!! –
   È SCOPPIATA UNA MINA!":GOTO 330
```

```
270 IF (minex=tankx) AND (miney=
   tanky) THEN PRINT TAB(minex,miney)"!"
   : PRINT TAB(7,7)"BOOM! –
   È SCOPPIATA UNA MINA!":GOTO 330
```

```
270 IF mx=tx AND my=
   ty THEN PRINT AT my,mx;"!"
   : PRINT AT 8,3;"BOOM!! –
   È SCOPPIATA UNA MINA":GOTO 330
```

```
270 M=PO THEN POKE 1024+PO,32:
   PRINT Z$;"▣BOOM!! –
   È SCOPPIATA UNA MINA!":GOTO 330
```



La linea 280 è quella importante: controlla se il carro armato e il paracadutista sono sullo stesso punto dello schermo. In tal caso, il punteggio aumenta di 1.

La linea 40 pone il punteggio a zero prima che il gioco inizi e la linea 330 visualizza il punteggio. Cambiando la fine della linea 270, il computer visualizza il putenggio dopo che è scoppiata una mina.

Il giocatore ha una serie di paracadusti da salvare. Quando uno è in salvo, un altro si lancia dal cielo. Il gioco termina quando una mina esplode, perché il carro armato è finito sullo stesso punto dello schermo da essa occupato.

## PUNTEGGI MASSIMI

Aggiungere al gioco un modulo per i punteggi massimi non è difficile. Tutto ciò di cui si ha bisogno è una variabile per il punteggio massimo (HS) e un modo per incrementarla quando il punteggio più alto viene superato, oltre a una routine per visualizzarla sullo schermo.

Ecco le linee da aggiungere per ottenere il modulo per il punteggio più alto:

```
30 LET HS=0
350 IF S>HS THEN LET HS=S
370 PRINT@ 424,"PUNTEGGIO MASSIMO=";
   HS;
```

```
30 LET HS=0
350 IF S>HS THEN LET HS=S
370 PRINT TAB
   (12,20) "PUNTEGGIO MASSIMO=";HS
```

```
30 LET hs=0
350 IF s>hs THEN LET hs=s
370 PRINT AT 18,8;"PUNTEGGIO MASSIMO
   =";hs
```

```
30 LET HS=0
350 IF S>HS THEN LET HS=S
370 PRINT "▣□□□□□□□□□
   PUNTEGGIO MASSIMO=";HS
```

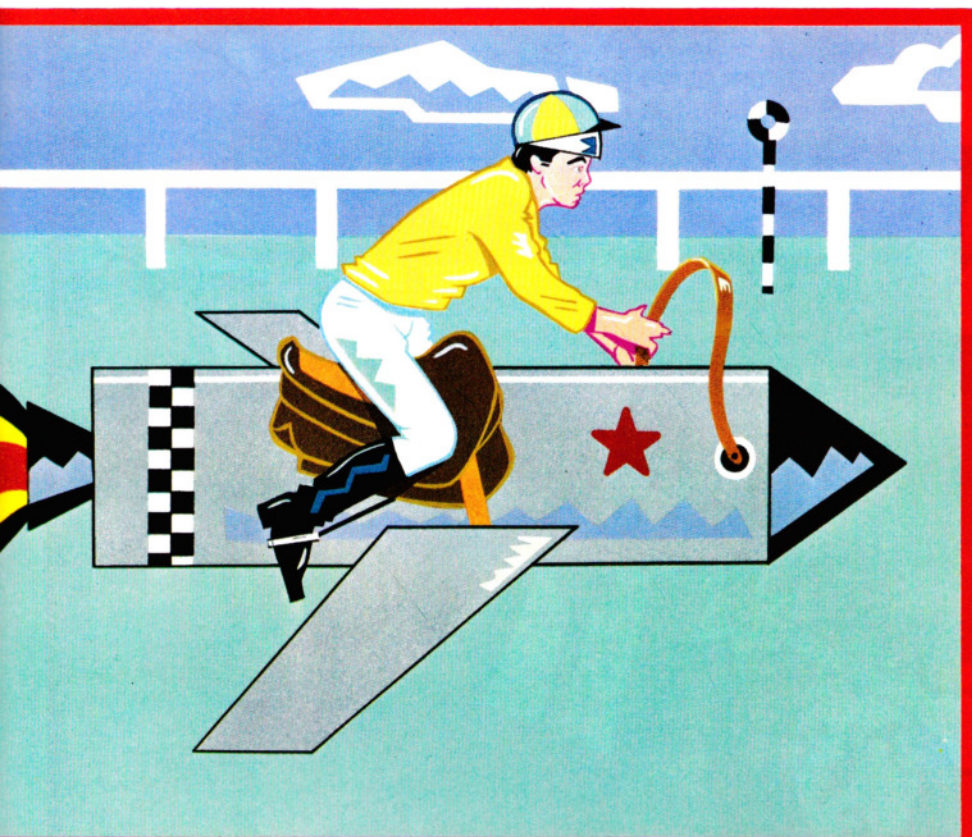
Dapprima, il punteggio massimo va posto al livello più basso possibile, così la linea 30 pone HS a zero. Quando il gioco si è fermato, la linea 350 confronta l'ultimo punteggio (S) con il punteggio massimo (HS). Se l'ultimo punteggio ottenuto è più alto del massimo, allora tale valore viene assegnato a HS. Infine, la linea 370 visualizza sullo schermo il punteggio massimo.

Forse sembrerà che queste linee bastino per aggiungere un controllo del massimo punteggio, ma, purtroppo non è così.

Ogni volta che si dà il RUN, il computer "dimentica" il precedente valore di HS, assieme ai valori di tutte le altre variabili. Per conservare il valore di HS si aggiungono queste poche linee, già viste in precedenza (pagina 35):

```
390 FOR F=1 TO 1000:NEXT F
400 PRINT@ 130
410 PRINT@ 135,"UN ALTRO GIOCO (S/N)?";
420 LET K$=INKEY$:IF K$=
   "" THEN GOTO 420
430 IF K$="S" THEN GOTO 40
440 IF K$="N" THEN END
450 GOTO 420
```

```
390 FOR F=1 TO 4000: NEXT F
400 PRINT TAB(7,7) STRING$(26,"□")
410 PRINT TAB(11,8)"UN ALTRO GIOCO
   (S/N)?"
420 LET K$=GET$
430 IF K$="S" THEN GOTO 40
440 IF K$="N" THEN END
450 GOTO 420
```

```
390 PAUSE 100
400 PRINT AT 8,3; TAB31
410 PRINT AT 8,7;"UN ALTRO GIOCO (S/N)?"
```

```
420 LET a$=INKEY$
430 IF a$="s" THE GOTO 40
440 IF a$="n" THEN STOP
450 GOTO 420
```

```
390 FOR F=1 TO 1000:NEXT F
410 PRINT "▣□□□□□□□□□
   UN ALTRO GIOCO (S/N)?";
420 GET K$:IF K$="" THEN GOTO 420
430 IF K$="S" THEN GOTO 40
440 IF K$="N" THEN PRINT "♥": POKE
   650,0:END
450 GOTO 420
```

Ecco le funzioni della linee aggiunte.

C'è una piccola pausa, ottenuta con un ciclo FOR ... NEXT alla linea 390 sull'Acorn, sul Commodore, sul Dragon e sul Tandy e con una PAUSE nel programma Spectrum. La linea 400 nel programma Acorn, Dragon, Tandy e Spectrum sgombra una porzione di schermo per il messaggio "UN ALTRO GIOCO? (Y/N)", visualizzato dalla linea 410.

La routine di richiesta è compresa tra le linee 410 e 450; la linea 430 fa riavviare il programma dalla linea 40 se si preme [Y] e la linea 440 ferma il programma se si preme [N]. La linea 450 si accerta che ogni altro tasto venga ignorato.

Adesso non occorre più impartire un RUN ogni volta che si vuol fare un'altra partita e, al tempo stesso, si conserva il valore di HS. Tuttavia, anche riuscendo a riavviare il programma senza scrivere RUN, al prossimo LOAD il programma avrà perso il valore di HS.

## CONTROLLO DEL TEMPO

Così com'è, il gioco dipende troppo dalla fortuna: il giocatore va avanti finché non si imbatte nella mina nascosta.

Si può introdurre nel gioco un elemento in più, trasformandolo in una gara contro il tempo. Con queste aggiunte si può misurare quanto ci vuole per salvare dieci paracadutisti.

```
80 TIMER=0
290 IF S<10 AND X=PO THEN GOTO 90
300 IF S=10 THEN GOTO 320
320 LET T=TIMER
340 IF S=10 THEN PRINT@ 327,"IN";T/50;
   "SECONDI"
```

```
10 LET T=0
80 TIME=0
290 IF S<10 AND (parax=
   tankx) AND (paray=tanky) THEN
   GOTO 90
300 IF S=10 THEN GOTO 320
320 LET T=TIME
340 IF S=10 THEN PRINT TAB(12,16)"N□";
   T/100;"□SECONDI"
```

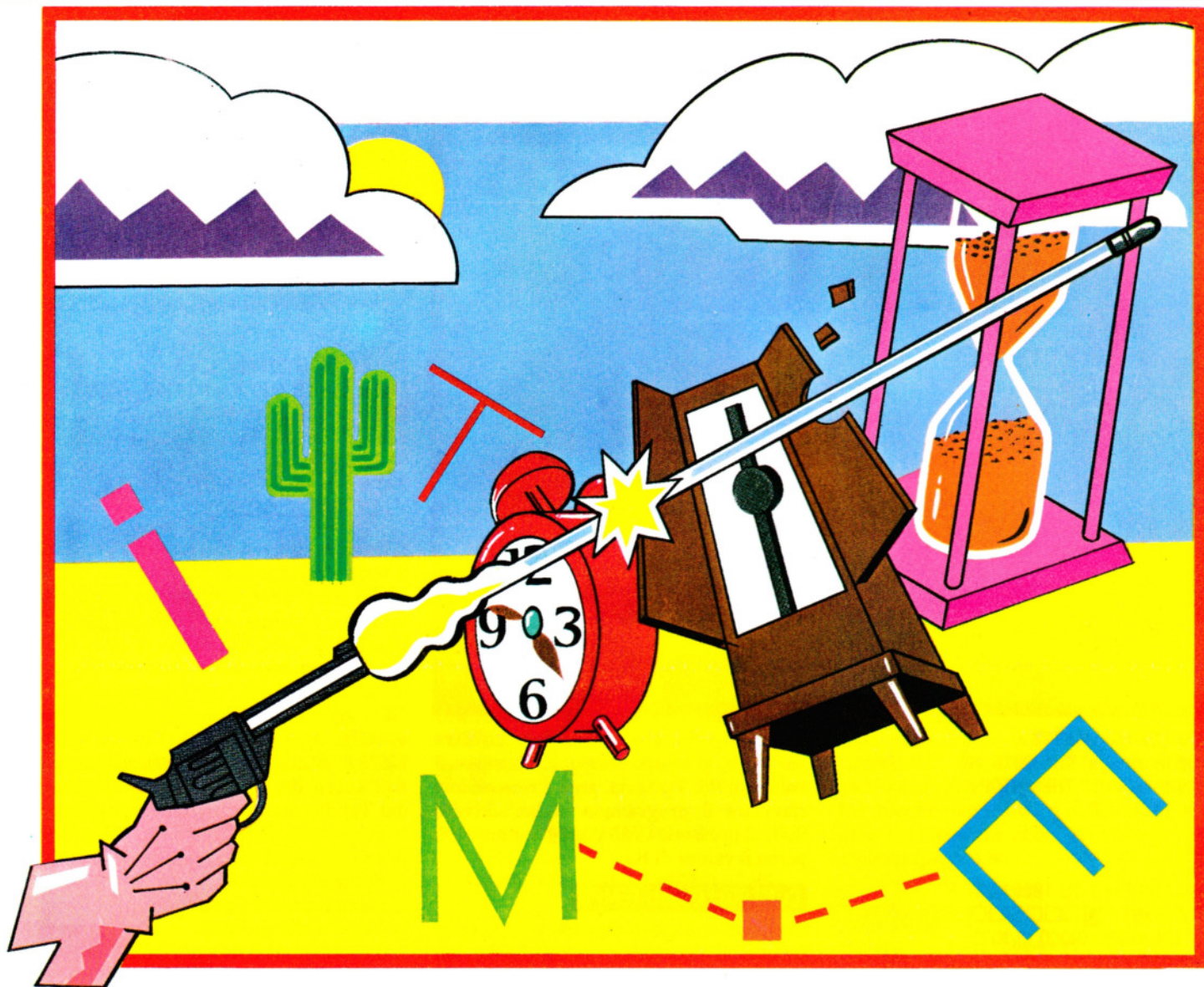
```
10 LET T=0
80 POKE 23672,0: POKE 23673,0
290 IF s<10 AND px=tx AND py=
   ty THEN GOTO 90
300 IF s=10 THEN GOTO 320
320 LET t=PEEK 23672+256*PEEK 23673
340 IF s=10 THEN PRINT AT 16,8;"IN□";
   t/50;"□SECONDI"
```

```
80 LET TI$="000000":L=5
290 IF S<10 AND X=PO THEN GOTO 90
300 IF S=10 THEN GOTO 320
320 LET T=VAL(RIGHT$(TI$,2))+60*VAL
   (MID$(TI$,3,2))
340 IF S=10 THENPRINT"▣□□□□□□
   □□IN";T;"□SECONDI"
```

Il timer in ogni macchina scorre per tutto il tempo che il computer è acceso. Tutto quello che si deve fare, per avviare il contatempo, è azzerarlo all'inizio del gioco: la linea 80 si 'occupa' proprio di questo. Si noti che, per regolare il timer dell'Acorn, si digita TIME=0, sul Commodore LET TI$="000000" sul Dragon/Tandy TIMER=0 e sullo Spectrum POKE 23672,0: POKE 23673,0. È chiaro quel che fanno le linee dell'Acorn, del Commodore, del Dragon e del Tandy, ma i possessori di uno Spectrum saranno un po' confusi. Le due POKE pongono i valori di due particolari locazioni di memoria a zero. La prima locazione di memoria conta fino a 255 prima di "traboccare" nella seconda.

Il contatempo è *fermato* dalla linea 320. In realtà, l'orologio non può venir fermato, quindi si fa in modo che la macchina "ricordi" qual era la lettura in un momento particolare (ad esempio, quando due posizioni coincidono sullo schermo). In questi programmi, la lettura del timer si chiama T: nell'Acorn LET T=TIME, nel Commodore LET T=VAL(RIGHT$(TI$,2)+60*VAL(MID$(TI$,3,2,)), nel Dragon/Tandy LET T=TIMER e nello Spectrum LET t=PEEK 23672+256*PEEK 23673.

Di nuovo, il metodo dello Spectrum può essere un po' oscuro. Vengono lette le due locazioni di memoria che prima, nel programma, erano poste a zero. La locazione 23672, come ogni locazione di memoria in un micro a 8 bit, può contenere numeri interi tra 0 e 255. La locazione 23673 scatta di 1 ogni volta che la locazione 23672 trabocca.

Così, per calcolare il tempo totale conteggiato dal timer dello Spectrum, si deve

moltiplicare il valore della locazione 23673 per 256 e aggiungerlo al valore contenuto nella locazione 23672.

Ora si può capire perché t corrisponde a PEEK 23672+256*PEEK 23673. Parleremo ampiamente di PEEK e POKE più avanti.

L'orologio del Commodore funziona aggiornando una stringa di sei caratteri predisposta dalla linea 80. Partendo da destra, la stringa contiene due cifre per le ore, due per i minuti e due per i secondi. La linea 320 valuta la stringa per avere una lettura in secondi. Il numero di minuti trascorsi è moltiplicato per 60 e aggiunto al numero di secondi: MID$(TI$,3,2,) esamina il numero dei minuti, e RIGHT$(TI$,2) il numero di secondi.

L'orologio deve fermarsi quando il giocatore ha salvato dieci paracadutisti, così la linea 300 controlla se dieci paracadutisti sono in salvo e poi salta alla 320 che *ferma* il timer. Nel programma Commodore, la linea 340 visualizza il tempo quando vengono salvati dieci paracadutisti.

Se un paracadutista riesce a salvarsi e il numero totale dei 'salvi' è minore di dieci, la linea 290 fa partire un altro lancio.

La linea 340 visualizza il tempo impiegato per salvare i dieci paracadutisti, ma *soltanto* in caso di successo. La lettura del timer è divisa per 50 nel Dragon, nel Tandy e nello Spectrum e per 100 nell'Acorn.

## TEMPO MINIMO

Come prima si è aggiunto al gioco il punteggio massimo, adesso si può aggiungere un modulo per il tempo minimo. In questo gioco si può memorizzare il tempo più breve impiegato a salvare i dieci paracadutisti, anche se il principio può essere applicato a qualsiasi misurazione di tempo si voglia.

Queste sono le linee da aggiungere:

```
20 LET LT=999999
360 IF T<LT AND S=10 THEN LET LT=T
380 PRINT@ 452,"MIGLIOR TEMPO=";
    LT/50;"SECONDI"
```

```
20 LET LT=999999
360 IF T<LT AND S=10 THEN LT=T
380 PRINT TAB(12,21) "MIGLIOR TEMPO=";
    LT/100;"□SECONDI"
```

```
20 LET lt=999999
360 IF t<lt AND s=10 THEN LET lt=t
380 PRINT AT 21,4;"MIGLIOR TEMPO=";
```

```
    lt/50; "□ SECONDI"
```

```
20 LET LT=999999
360 IF T<LT AND S=10 THEN LET LT=T
380 PRINT"■□□□□□□□□□
    MIGLIOR TEMPO=";LT;"□SECONDI"
```

Se, al punteggio massimo, inizialmente viene assegnato un valore bassissimo (zero!), al tempo minimo, invece, si assegna un tempo minimo ridicolmente lungo. La linea 20 pone la variabile del tempo minimo (LT o lt) al valore di 999999.

La linea 360 continua a confrontare l'ultimo tempo con il tempo minimo. Se l'ultimo tempo è minore del tempo minimo *e* dieci paracadutisti sono al sicuro, allora il tempo minimo viene modificato in modo da coincidere con nuovo miglior tempo.

Infine, la linea 380 visualizza il tempo minimo in secondi. La variabile del tempo minimo è divisa per 50 o 100 per ottenere un valore in secondi.

Anche quando si usa un modulo per il tempo minimo in un gioco, occorre una routine per la richiesta di un altro gioco, altrimenti si perderebbe il valore del tempo minimo ogni volta che si dà un nuovo RUN al programma.

## IL TEMPO E LA TASTIERA

Abbiamo visto come controllare il timer della macchina, dall'interno di un programma, verificando la posizione di due oggetti sullo schermo.

Un altro modo per *fermare* l'orologio è usare la tastiera.

A pagina 54 viene mostrato come usare INKEY$ o GET$. Possono essere usati, con la stessa facilità, tanto per avviare o fermare il timer, quanto per controllare oggetti sullo schermo.

Ecco un gioco del tipo "Chi spara per primo" nel quale è mostrato come si possa usare la tastiera per fermare il timer:

```
20 CLS
30 LET N=RND(900)
40 FOR F=0 TO N
50 NEXT F
60 PRINT@ 269, "SPARA!"
70 TIMER=0
80 IF INKEY$=" " THEN GOTO 80
90 LET T=TIMER
100 PRINT@ 269,"BANG!!"
110 FOR F=1 TO 300
120 NEXT F
130 LET M=RND(25)
140 IF T < M THEN PRINT@ 264,
    "SEI SOPRAVVISSUTO"
150 IF T>M THEN PRINT@ 266,
    "SEI STATO COLPITO"
160 IF T=M THEN PRINT@ 264,
    "SIETE AMBEDUE STECCHITI!"
```

```
20 CLS
30 LET N=RND(2000)
40 FOR delay=0 TO N
50 NEXT delay
60 PRINT TAB(17,13) "SPARA!"
70 TIME=0
80 K$=GET$
90 LET T=TIME
100 PRINT TAB(17,13) "BANG!!"
110 FOR delay=1 TO 1000
120 NEXT delay
130 LET M=RND(40)
140 IF T<M THEN PRINT TAB(12,13) "SEI
    SOPRAVVISSUTO"
150 IF T>M THEN PRINT TAB(14,13) "SEI
    STATO COLPITO"
160 IF T=M THEN PRINT TAB(12,13)
    "SIETE AMBEDUE STECCHITI!"
```

```
20 CLS
30 LET n=INT (RND*400)+1
40 PAUSE n
60 PRINT AT 11,14; "SPARA!"
70 POKE 23672,0: POKE 23673,0
80 IF INKEY$=" " THEN GOTO 80
90 LET T=PEEK 23672+256*PEEK23673
100 PRINT AT 11,14; "BANG!!"
110 PAUSE 50
130 LET m=INT (RND*35)+1
140 IF t>m THEN PRINT AT 11,9; "SEI
    SOPRAVVISSUTO"
150 IF t>m THEN PRINT AT 11,11; "SEI
    STATO COLPITO"
160 IF t=m THEN PRINT AT 11,9; "SIETE
    AMBEDUE STECCHITI!"
```

**D+R**

**C'è un limite al periodo di tempo che si può misurare?**

Sì, anche se in genere è così lungo da essere, in pratica, ininfluente. Negli home computer di solito il timer ha lo stesso 'passo' e il fattore limitante è il numero di impulsi che il computer riesce a ricordare.

Dragon, Tandy e Commodore possono trattenere fino a due byte (65.535), equivalenti a circa 22 minuti. Lo Spectrum si spinge fino a tre byte, ossia quasi quattro giorni. Il BBC e l'Electron conservano il tempo in 4 byte e si possono predisporre per contare quasi 350 anni!

```
20 PRINT "■"
30 LET N=INT(RND(1)*900)+1
40 FOR F=0 TO N
50 NEXT F
60 PRINT "■■■■■■■";TAB(16);
    "SPARA!"
70 LET TI$="000000":POKE 198,0
80 GET K$:IF K$=" "THEN GOTO 80
90 LET T=TI
100 PRINT "■■■■■■■";TAB(16);
    "BANG!!"
110 FOR F=1 TO 300
120 NEXT F
130 LET M=INT(RND(1)*35)+1
140 IF T<M THEN PRINT
    "■■■■■■■■■"; TAB(51)
    "SEI SOPRAVVISSUTO"
150 IF T>M THEN PRINT
    "■■■■■■■■■" TAB(53)
    "SEI STATO COLPITO"
160 IF T=M THEN PRINT
    "■■■■■■■■■" TAB(50)
    "SIETE AMBEDUE STECCHITI!"
```

Il programma visualizza 'SPARA!' e il giocatore deve premere al più presto un tasto. La reazione è calcolata dal momento in cui appare il messaggio e il giocatore è avvertito se ha vinto lui o il computer.

Il programma è molto semplice. Dopo che la linea 20 ha ripulito lo schermo viene introdotta una pausa, di lunghezza casuale, dalle linee 30-50. La linea 60 visualizza 'SPARA!' e il timer è immediatamente avviato dalla linea 70. La linea 80 fa attendere la macchina, prima di continuare, finché non viene premuto un tasto. La linea è usata esattamente come spiegato in precedenza a pagina 54. Appena viene premuto un qualsiasi tasto, il programma procede alla linea 90 che ferma il timer 'invitandolo' a leggere T. La linea 100 visualizza un 'BANG!'.

C'è una pausa introdotta dalle linee 110 e 120 (nello Spectrum solo dalla 110), prima che la macchina scelga il momento dell'estrazione (linea 130). La macchina ha due variabili: il tempo del giocatore T e il 'tempo' della macchina M. Le linee da 140 a 160 confrontano i due tempi e visualizzano il risultato della sparatoria.

Sugli Acorn non si provi a usare INKEY per provocare i ritardi alle linee 30 e 110 come fatto nei precedenti programmi di animazione. In questa circostanza, premendo un tasto, il ritardo viene interrotto, quindi il risultato sarebbe quello di far concludere il gioco in un lampo!

# TUTTO SU READ E DATA

**Far leggere al computer un gruppo di frasi DATA ci evita di digitare programmi lunghi e lenti da far girare. È una tecnica utile per qualsiasi applicazione: dalla grafica ai semplici elenchi.**

La versatilità dei computer proviene in gran parte dall'uso delle variabili, alle quali è facilissimo assegnare valori. Per esempio, si scrive semplicemente LET X=5. Ma, talvolta, la quantità di informazioni da usare non è agevolmente controllabile: ecco quando tornano utili le istruzioni DATA e i comandi associati READ e RESTORE. Ricordiamo che quest'ultimo non è disponibile nel BASIC dello ZX81.

La parola DATA (dal latino *datum*) ha, in BASIC, un significato molto preciso: è, in effetti, un'istruzione. Non va confusa con il generico termine "dati" usato per indicare le più svariate cose, ad esempio un programma, una routine in linguaggio macchina oppure una matrice archiviati su nastro o su dischetto. Il primo metodo nel quale ci si imbatte, per assegnare un valore a una variabile (sia essa numerica o alfanumerica) prevede l'impiego di una INPUT. Ma questo sistema è utile soltanto quando l'informazione deve essere fornita al computer dall'opertore, ogni volta che viene eseguito il programma. Spesso, però, esistono valori fissi, che non occorre inserire o cambiare ogni volta: essi possono essere, quindi, incorporati una volta per tutte nel programma. Potremmo adoperare un'interminabile serie di LET, ma ciò è particolarmente laborioso e rallenta l'esecuzione del programma.

Ecco un esempio in cui, per memorizzare una lista di parole, viene usata una serie di LET:

```
10 LET A$="GIORNO"
20 LET B$="SETTIMANA"
30 LET C$="MESE"
40 LET D$="ANNO"
50 PRINT A$,B$,C$,D$
```

Un metodo più rapido ed efficente, però, consiste nell'uso di frasi DATA:

```
10 READ A$, B$,C$,D$
20 PRINT A$,B$,C$,D$
100 DATA GIORNO,SETTIMANA,MESE,ANNO
```

Se ora volessimo usare non quattro, ma cinquanta parole diverse, il primo programma richiederebbe altre 46 istruzioni, il secondo solo una o due, secondo il computer e la dimensione degli elementi.

Si ricordi che, sullo Spectrum, gli elementi della frase DATA devono essere racchiusi da apici. Così la linea 100 risulta:

```
100 DATA "GIORNO","SETTIMANA","MESE",
"ANNO"
```

## COME OPERANO DATA E READ

Come operano in pratica le istruzioni READ e DATA? Quando il computer incontra un'istruzione READ, scorre il programma finché trova la prima DATA. Quindi assegna il primo elemento di DATA alla variabile che segue l'istruzione READ.

Così, nel programma precedente, la stringa "GIORNO" è assegnata alla variabile A$, "SETTIMANA" a B$ e via dicendo.

Le frasi DATA compaiono, generalmente, nella parte finale di un programma, dove non intralciano il cammino, mentre il resto del programma viene scritto e "ripulito". In effetti, però, possono essere situate ovunque, poiché il computer le ignora finché una READ non ne legge il contenuto. Il seguente programma funzionerà perfettamente:

```
10 DATA GERMANIA
20 READ A$,B$
30 DATA, FRANCIA,ITALIA,SPAGNA
40 READ C$,D$
```

È ovvio che il programma risulta più leggibile se le DATA sono raggruppate tutte assieme. C'è una sola regola da osservare: le DATA devono comparire nell'ordine in cui il computer le leggerà in seguito.

Il numero di elementi che una singola frase DATA può contenere dipende dalla lunghezza massima di linea di ogni computer. Appena una linea è piena, se ne inizi tranquillamente un'altra: il computer le leggerà con ordine.

- COME FUNZIONANO **READ** E **DATA**
- I DIVERSI TIPI DI **DATA**
- COME EVITARE I PROBLEMI
- USARE **DATA** PER CREARE UN SEMPLICE ELENCO
- L'ISTRUZIONE **RESTORE**
- I DIVERSI IMPIEGHI DI **DATA**
- ESTRAZIONE DA UNA FRASE **DATA**
- COME ESEGUIRE SEMPLICI DISEGNI CON FRASI **DATA**

## DIVERSI TIPI DI DATA

Finora abbiamo usato frasi DATA contenenti stringhe, ma il discorso vale anche per valori numerici. Nei computer Spectrum ed Acorn, si possono addirittura usare variabili e anche funzioni, cosicché potremmo incontrare una linea di questo tipo:

```
DATA PARIGI, 256,a*5
```

(Sullo Spectrum, ripetiamo, "PARIGI" avrebbe richiesto l'uso di doppi apici).

Le variabili, nel comando READ, devono essere disposte seguendo lo stesso ordine dei corrispondenti elementi delle DATA. La linea di DATA appena vista, quindi, potrebbe essere letta con il comando:

```
READ A$,N,X
```

Si noti che la prima variabile è una variabile *stringa* per accoppiarsi con la prima voce della DATA. Le due seguenti invece sono variabili *numeriche* in quanto la READ, adesso, deve leggere numeri.

## PROBLEMI CON DATA

Non è raro commettere errori nell'immissione di frasi DATA. I principali problemi sorgono quando non si hanno abbastanza elementi da leggere o si usa la READ in modo scorretto. Se avessimo erroneamente digitato la precedente linea di DATA, scrivendo DATA 256, PARIGI,a*5, allora in A$ finirebbe '256'. Benché scorretto, non verrebbe rifiutato (salvo dallo Spectrum).

La cosa è diversa quando il computer tenta di inserire la stringa 'PARIGI' nella variabile N. Il computer può reagire in due modi: segnalandoci che la frase DATA è stata formulata male, emettendo un messaggio d'errore del tipo 'type mismatch' o 'bad data'. In alternativa, il computer potrebbe concludere che PARIGI è il nome di variabile che non ha precedentemente incontrato e potrebbe segnalarci ciò con un 'variable not found' (variabile non trovata).

Un altro comune errore consiste nell'inserire elementi insufficienti nelle DATA. Ciò può facilmente accadere se il comando di lettura READ è posto all'interno di un ciclo, come nel programma successivo. Basta un errore nei valori del ciclo, oppure l'omissione di un elemento nella DATA e il computer cercherà di leggere più dati di quelli contenuti nella frase DATA.

Il risultato è l'interruzione del programma con un messaggio del tipo 'out of data' (dati esauriti).

## UN SEMPLICE ELENCO

Ecco un programma che usa un ciclo per leggere frasi DATA. È un elementarissimo programma per rubrica telefonica. Si inserisce il nome della persona e il computer ne visualizza il numero telefonico.

```
5 CLS
10 PRINT TAB(10,2) "RUBRICA TELEFONICA"
20 INPUT TAB(5,10) "IMMETTERE IL
   NOME□",R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT TAB(5,12) "IL
   NUMERO DI□";N$; "È:";T$:END
60 IF N$="FINE" THEN PRINT TAB(5,12)
   R$;"□NON È NELL'ELENCO"
70 NEXT J
500 DATA GIANNI,02-43299,MARZIA,051-
   845003,MARCO,06-933921,GIULIA,010-
   4530199,ANTONIO,055-234502,FINE,FINE
```

```
5 CLS
10 PRINT@ 71,"RUBRICA TELEFONICA"
12 PRINT
20 INPUT "IMMETTERE IL NOME";R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT@ 320,"IL
   NUMERO DI□";N$; "È:";T$:END
60 IF N$="FINE" THEN PRINT@ 320,
   R$;"□NON È NELL'ELENCO"
70 NEXT J
500 DATA GIANNI,02-43299,MARZIA,051-
   845003,MARCO,06-933921,GULIA,010-
   4530199,ANTONIO,055-234502,FINE,FINE
```

```
5 PRINT "♥"
10 PRINT TAB(10)"RUBRICATELEFONICA"
20 INPUT "■IMMETTERE IL NOME";R$
25 PRINT
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT "IL NUMERO
   DI□"; N$;" È :"; T$: END
60 IF N$="FINE" THEN PRINT R$; "□NON
   È NELL'ELENCO"
70 NEXT J
500 DATA GIANNI,02-43299,MARZIA,051-
   845003,MARCO,06-933921,GIULIA,010-
   4530199,ANTONIO,055-234502,FINE,FINE
```

```
5 CLS
10 PRINT AT 2,6;"RUBRICA TELEFONICA"
20 INPUT "IMMETTERE IL NOME";R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT AT 10,1;"IL
   NUMERO DI□";N$; "È:";T$:STOP
60 IF N$="FINE" THEN PRINT AT 10,3; R$;
   "□NON È NELL'ELENCO"
```

```
70 NEXT J
500 DATA "GIANNI","02-43299","MARZIA",
    "051-845003","MARCO","06-933921",
    "GIULIA","010-4530199","ANTONIO","055-
    234502","FINE""FINE"
```

Si digiti il programma, usando però i nomi e numeri di telefono dei propri amici, nella linea di DATA. Vi si possono inserire quante voci si vogliono, purché si modifichi il numeratore del ciclo, alla linea 30.

Gli ultimi due elementi delle DATA sono FINE, FINE. Questo perché la linea 60 possa controllare se si è raggiunta la fine dell'elenco. FINE (sullo Spectrum, "FINE") va ripetuto due volte, se no avremmo un errore 'out of data', in quanto la linea 40 legge *due* elementi di DATA alla volta.

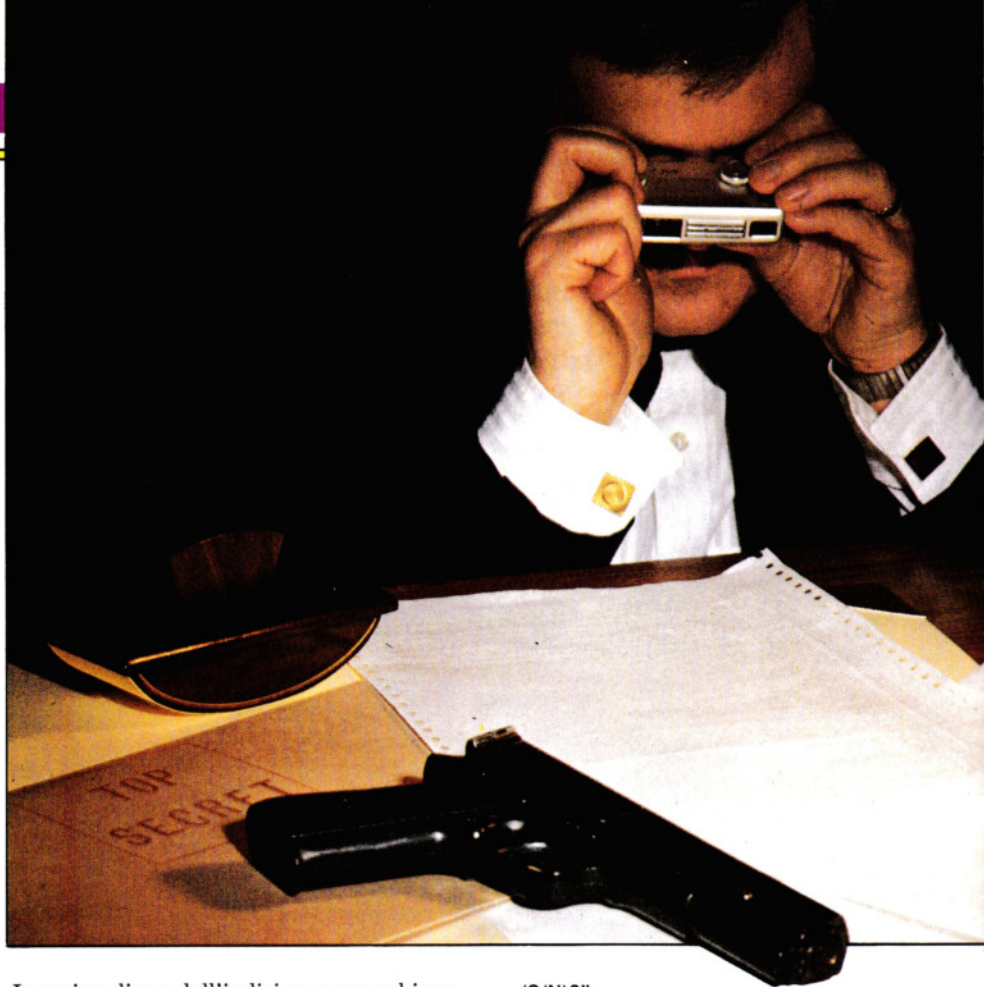
Si noti anche che il numero di telefono viene letto come stringa. Ciò permette di usare segni di interpunzione che un numero non potrebbe avere. Le stringhe nella DATA possono contenere spazi, se si vuole, ma non virgole. Infatti, la virgola è il carattere separatore tra i vari elementi e il computer si confonderebbe! Dovendo usare una virgola all'interno di una stringa, la si racchiuda tra doppi apici. Ciò sarebbe necessario in una frase DATA come:

```
10 READ N$,A$,B$,C$
20 DATA ALBERTO STERNI,"VIALE FOSCOLO,
    123", ROMA,00100
```

**Teniamo il bandolo delle DATA**

A qualsiasi programma si riferiscano, le frasi DATA hanno un elemento in comune: occorre molto tempo per definirle e per trascriverle. Non parliamo poi di decifrarle o modificarle dopo mesi che non si adopera un programma pieno di DATA.

Il tempo occorrente ad organizzare con metodo le frasi DATA è ben speso, perché facilita enormemente eventuali interventi successivi.

Il miglior criterio da seguire, benché esso dipenda in parte dal contenuto stesso delle DATA, è quello di incolonnare opportunamente i valori e di corredare il tutto con adeguate linee di commento (REM) ricche di spiegazioni sul formato e sul significato dei vari elementi.

La prima linea dell'indirizzo va racchiusa tra apici, causa la virgola dopo il numero 23. (Nello Spectrum ciò andrebbe fatto comunque, per tutte le stringhe).

## L'USO DI RESTORE

Con i metodi esposti finora, un programma può leggere le frasi DATA una sola volta, a meno che non si impartisca un nuovo RUN. Per rileggere più volte una frase DATA si usa l'istruzione RESTORE.

Supponiamo che si desiderino consultare più numeri telefonici. La cosa più ovvia sembra prevedere una routine del tipo "Un'altra volta (S/N)?". Si aggiungano le seguenti linee, dopo aver sostituito la END o la STOP della linea 50 con una GOTO 80:

```
80 PRINT TAB(2,20)"VUOI UN ALTRO
    NUMERO (S/N)?"
90 K$=GET$
100 IF K$="S" THEN GOTO 5
110 END
```

```
80 PRINT AT 12,0;"VUOI UN ALTRO NUMERO
    S/N?"
90 PAUSE 0
100 IF INKEY$="s" THEN GOTO 5
110 IF INKEY$="n" THEN GOTO 2000
```

```
80 PRINT@ 32, "VUOI UN ALTRO NUMERO
    (S/N)?"
90 K$=INKEY$:IF K$=" "THEN GOTO 90
100 IF K$="S" THEN GOTO 5
110 END
```

```
80 PRINT TAB(4)"VUOI UN ALTRO NUMERO
    (S/N)?"
100 IF K$="S" THEN GOTO 5
110 END
```

Proviamo adesso a lanciare il programma. La prima volta lavora bene. Ma quando si preme un tasto per un altro *giro*, il programma si blocca per mancanza di DATA. Ciò avviene perché la lettura, dopo il primo RUN, ha esaurito gli elementi da leggere. Ma c'è un rimedio; si aggiunga:

```
15 RESTORE
```

Stavolta, il programma continua a funzionare regolarmente: il comando RESTORE, infatti, istruisce il computer a ritornare all'inizio della lista di DATA.

Durante la stesura di un programma, è buona norma inserire l'istruzione RESTORE tra le prime istruzioni: ciò impedisce di rimanere senza più DATA dopo ripetuti RUN di prova. Se, nella versione definitiva del programma, RESTORE va tolta, lo si annoti con una frase REM.

Usando RESTORE si può rileggere una serie di frasi DATA per quante volte vogliamo. Ciò è particolarmente utile quan-

do vanno ripetute per diverse volte cose come etichette o intestazioni di tabelle. Per esempio, si possono mettere i mesi dell'anno in frasi DATA, per il programma di un calendario che ne fa un uso ripetitivo in tabelle o grafici. L'istruzione RESTORE è comoda anche, come mostrato nel programma della rubrica telefonica, quando si vuol continuare a scorrere una lista alla ricerca di un certo elemento.

## DIVERSI IMPIEGHI DI DATA

Le frasi DATA sono estremamente utili in tutti i tipi di programmi. Sono già state usate per la grafica definita dall'utente (pagine 8-15) e per disegnare labirinti (pagine 68-74), ma si possono adoperare anche per far leggere al computer parametri per effetti musicali o sonori.

Spesso, nei giochi di avventure, molte linee di programma contengono, in frasi DATA, il testo necessario al gioco. Nei quiz si possono raccogliere tutte le domande e risposte in linee di DATA. E nei giochi da 'arcade' scritti in BASIC si usano DATA per definire sia personaggi che ambienti.

I programmatori esperti usano DATA per programmi in linguaggio assembly o in codice macchina. Un programma di questo tipo può consistere in un breve ciclo di lettura delle DATA, associato al comando POKE. Se adoperate con attenzione, le DATA possono diventare un efficace strumento di programmazione.

## PUNTATORI PER LA RESTORE

Finora, l'unico problema con DATA è stato richiamare sempre l'informazione nella stessa sequenza, a partire dallo stesso punto. Usando RESTORE è possibile tornare all'inizio di una lista anche se non se ne è raggiunta la fine. Come saltare però nel *mezzo* di una lista?

Sui computer Acorn e sullo Spectrum, esiste un sistema: invece di usare una sola serie di DATA, se ne possono in effetti usare diverse, dirigendo il computer verso quella giusta. Al programma di grafica (riportato più avanti) si aggiungano le seguenti linee:

```
15 INPUT "CASA <S>TRETTA OPPURE
   <L>ARGA",H$
16 IF H$="S" THEN RESTORE 2000
17 IF H$="L" THEN RESTORE 1000
18 IF H$<>"S" AND H$<>"L" THEN
   GOTO 15
2000 DATA 400,100,900,100,900,600,400,
   600,400,100
2010 DATA 500,200,550,200,550,400,500,
   400,500,200,750,200,800,200,800,400,
   750,400,750,200
2020 DATA 600,100,700,100,700,400,600,
   400,600,100
2030 DATA 300,600,1000,600,650,800,300,
   600
```

```
6 INPUT "PONTE ANTICO O MODERNO
  (A/M)?";y$
7 IF y$="a" THEN RESTORE 1000
8 IF y$="m" THEN RESTORE 2000
9 IF y$<>"a" AND y$<>"m" THEN
  GOTO 5
2000 DATA 83, 84, 85, 86, 87, 88, 89, 90,
  90, 90, 90, 90, 90, 90, 90, 89, 88, 87, 86,
  85, 84, 83
2010 DATA 42, 55, 63, 65, 63, 55, 42
```

I numeri che seguono le istruzioni RESTORE, noti come i *puntatori* per la RESTORE, indicano al computer da quale linea di DATA iniziare la lettura. Così, nel programma Acorn, premendo una [L] per una casa *larga*, il puntatore RESTORE diventa 1000 alla linea 17. Nel programma per lo Spectrum, la linea 7 ha un effetto simile.

Se si vuole che il computer vada proprio alla prima lista di DATA, non è necessario un puntatore: la RESTORE senza numero di linea corrisponde a una RESTORE con il numero di linea della prima DATA. La linea 17 del programma dell'Acorn poteva essere scritta IF H$="L" THEN RESTORE con identico effetto. I puntatori RESTORE sono molto utili nel programmare giochi. In un gioco di atterraggio, per esempio, si può scrivere il programma che verifichi se ci si è salvati oppure sfracellati al suolo. Le frasi DATA conterranno allora informazioni per emettere una fanfara di vittoria o il rumore di un esplosione e il puntatore RESTORE indicherà l'appropriata linea DATA da usare.

## USO DI DATA PER LA GRAFICA

Le frasi DATA sono molto utili anche nei programmi di grafica, per fissare le coordinate di disegni e diagrammi o per richiamare simboli grafici ROM. L'uso di DATA è ideale per forme irregolari come quelle nella pagina successiva, dato che servirebbero molte più linee di programma se si usassero singole istruzioni di visualizzazione o di disegno.

In qualche caso, però, le DATA non sono d'aiuto. È il caso in cui si tracciano forme molto regolari dove si possono calcolare le coordinate o la forma grafica.

Le frasi DATA nel programma successivo, sono state volontariamente suddivise in più linee, perché riesce difficile modificare frasi DATA non sono disposte con un certo ordine: si immagini di tornare su un programma dopo qualche tempo e di provare ad abbinare le voci della DATA ad ogni variabile. Si può, certo, ricostruire ogni operazione della READ, ma ciò è noioso se il programma è lungo. Conviene quindi suddividere bene ogni gruppo di istruzioni DATA e, se si ha sufficiente memoria, si usino REM in abbondanza per rendere le cose ancora più chiare.

Il seguente programma disegna una casa:

```
10 MODE 0
15 RESTORE
20 FOR N=1 TO 4
30 READ X,Y
40 MOVE X,Y
50 FOR M=1 TO 4
60 READ X,Y
70 DRAW X,Y
80 NEXT M
90 NEXT N
100 READ X,Y
110 MOVE X,Y
120 FOR P=1 TO 3
130 READ X,Y
140 DRAW X,Y
150 NEXT P
160 PRINT'' "PREMERE UN QUALSIASI
   TASTO PER CONTINUARE"
170 Q=GET
180 CLS: GOTO 10
1000 DATA 300,100,1000,100,1000,400,300,
   400,300,100
1010 DATA 400,200,500,200,500,300,400,
```

```
    300,400,200,800,200,900,200,900,300,
    800,300,800,200
1020 DATA 600,100,700,100,700,300,600,
    300,600,100
1030 DATA 200,400,1100,400,650,600,200,
    400
```

Nelle DATA sono raccolte le coordinate per le varie parti della casa. La linea 1000 dà il contorno principale, la linea 1010 le finestre, la linea 1020 la porta e la 1030 il tetto. La parte principale del programma predispone i cicli per la lettura dei pezzi corrispondenti di DATA e per il disegno delle linee. La linea 20 prepara il ciclo per disegnare quattro rettangoli (il contorno della casa, le due finestre e la porta), mentre la 50 prepara il ciclo per disegnare quattro lati a ogni rettangolo. Infine, le linee da 120 a 150 disegnano i tre lati del tetto.

Una volta disegnata la casa, viene chiesto se si vuole ripetere il programma: basta premere un qualsiasi tasto. Si noti la RESTORE alla linea 15, che permette al programma di rileggere la frasi DATA.

Questo programma usa READ ... DATA per disegnare un ponte. Digitando e avviando il programma un 'pezzo' alla volta, è più facile vedere cosa succede e controllare che la battitura sia corretta.

```
10 FOR t=74 TO 80 STEP 3
20 PLOT 35,t
30 DRAW 175,0,-2.5
40 NEXT t
```

Questa parte usa un ciclo FOR ... NEXT per porre tre punti vicini al lato sinistro dello schermo e poi disegnare una linea curva da ogni punto. Alla linea 30 ci si renderà conto che 175,0 significa "fino a un punto 175 pixel alla destra del pixel di partenza". Il −2.5 fa sì che la linea sia curva anziché retta.

```
100 FOR n=18 TO 39
110 READ a
120 PLOT n,45
130 DRAW 0,a
132 PLOT n+188,45
134 DRAW 0,a
140 NEXT n
1000 DATA 70,70,67,67,70,70,60,60,57,57,60,
    60,57,57,60,60,70,70,67,67,70,70
```

Questa è la sezione che disegna le torri. Il ciclo tra le linee 100 e 140, più il numero 45 (pixel dal basso) alle linee 120 e 132 traccia i punti alla base di ciascuna delle linee verticali che disegnano le torri.

Poi subentrano le linee 110 e 1000. La linea 110 dice al computer di leggere, alla linea 1000, l'altezza (ancora in pixel) di ciascuna delle 22 linee verticali. Così, la prima linea è 0,70 (ossia una verticale alta 70 pixel), la seconda linea è 0,70, la terza 0,67 ecc. Volendo osservare la progressiva costruzione del disegno, si inserisca una provvisoria 135 PAUSE 100 dopo la 134.

```
300 PLOT 0,75
310 DRAW 255,0,-0.1
320 PLOT 0,78
330 DRAW 255,0,-0.1
```

Queste linee disegnano la strada, con −0.1 che genera una lieve curva all'insù. E infine:

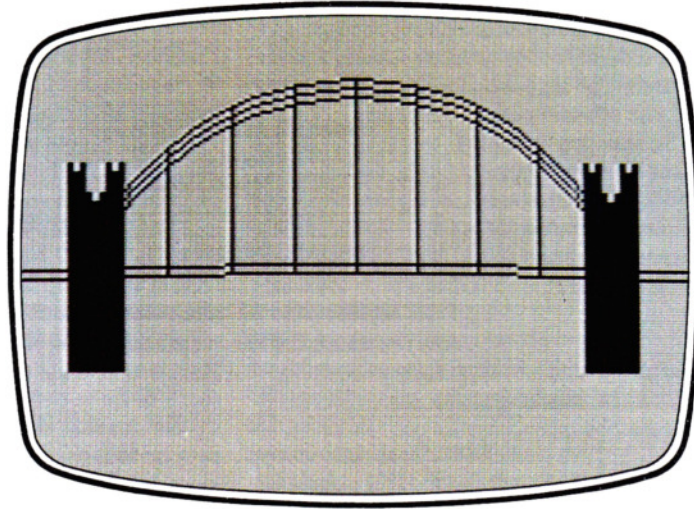
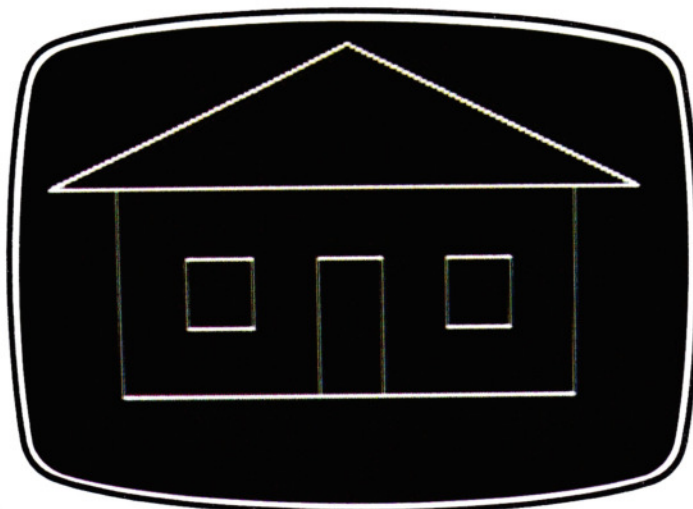
```
400 FOR r=62 TO 182 STEP 20
410 PLOT r,78
420 READ b
430 DRAW 0,b
440 NEXT r
1010 DATA 42,55,63,65,63,55,42
```

Queste linee producono i tiranti tra l'arco e la strada, con il ciclo FOR ... NEXT che assegna le posizioni di partenza dei tiranti, mentre le linee 420 e 1010 regolano la loro altezza. Se si vuole rieseguire automaticamente il programma, si aggiungano le seguenti linee:

```
5 CLS:RESTORE
450 GOTO 5
```

La linea 5 pulisce lo schermo e permette poi al programma di rileggere la DATA.

Questo programma usa una routine READ ... DATA per disegnare una casetta di campagagna sfruttando in parte i simboli grafici ROM del Commodore 64:

```
10 PRINT "♥"
20 POKE 53280,3:POKE 53281,5
30 FOR A=1 TO 7
40 FOR B=1 TO 14
50 READ X
60 POKE 1024+291+(A*40)+B,X
65 POKE 55296+291+(A*40)+B,7
70 NEXT B
75 NEXT A
500 DATA 32,32,32,32,32,32,32,32,32,32,108,
    160,123,32
505 DATA 32,32,233,160,223,32,32,32,32,
    118,160,160,160,116
510 DATA 32,233,160,160,160,223,32,92,32,
    160,160,160,160,160
520 DATA 32,102,102,102,102,102,102,160,160,
    223,108,160,102,160,120
530 DATA 32,102,91,102,172,102,102,91,
    102,32,32,102,32,32
540 DATA 32,102,102,102,160,102,102,102,
    102,32,32,102,32,32
550 DATA 99,99,99,99,99,99,99,99,99,99,99,
    99,99,99
```

1. Una casa disegnata con solo 4 frasi DATA (Acorn)

2. Un ponte usando PLOT, DRAW e qualche DATA (Spectrum)

Vengono usate sette linee di DATA: queste corrispondono alle sette righe dello schermo che fanno parte del ciclo FOR ... NEXT avviato alla linea 30. Le quattordici voci in ogni linea di DATA corrispondono alle quattordici colonne specificate nel ciclo FOR ... NEXT che ha inizio alla linea 40.

Dopo che il valore di X (un codice di visualizzazione) è stato letto dalla linea 50, viene depositato nella memoria del video con una POKE, come indicato alla linea 60. Una corrispondente POKE deve intervenire nella memoria colore per poter visualizzare il carattere: ciò si ottiene con la linea 65. In ambedue queste linee, il numero 291 fa partire il disegno da un punto situato al centro dello schermo, ma la posizione reale è dettata dai valori delle variabili A e B. Si digitino ora queste linee:

```
220 FOR K=1 TO 2000
230 NEXT K
240 PRINT "♥▚□ PREMERE QUALSIASI
   TASTO PER CONTINUARE"
250 GET K$:IF K$="" THEN GOTO 250
270 GOTO 10
```

Si dia un RUN e si risponda all'invito che segue alla pausa. Comparirà un messaggio d'errore 'out of data': la DATA è esaurita e non permette una seconda esecuzione (linea 50). Aggiungendo ora:

```
260 RESTORE
```

che fa tornare il computer all'inizio dell'istruzione DATA, il programma può essere eseguito quante volte si desidera.

Questo programma costruisce, a tappe, il disegno di un castello.

Si battano queste linee e si dia il RUN:

```
10 PCLEAR4
20 PMODE4,1
30 PCLS5
40 SCREEN1,1
50 READ SX,SY
60 LINE-(SX,SY),PSET
70 FOR K=1 TO 18
80 READ X,Y
90 LINE-(X,Y),PRESET
100 NEXT K
270 GOTO 270
500 DATA 64,160
510 DATA 64,60,32,60,48,40,64,60,32,60,32,
   160,110,160,110,120
520 DATA 152,120,152,160,228,160,228,60,
   212,40,196,60,228,60
530 DATA 196,60,196,160,196,74
```

Le linee da 10 a 40 preparano lo schermo per la grafica ad alta risoluzione. La linea 270 mantiene tale "modo" inserito. Le linee 50 e 60 sono insolite; le coordinate del punto di partenza del castello sono lette e una linea color nocciola è disegnata fino al punto di partenza su uno sfondo color nocciola. Talvolta è utile saper disegnare queste linee *invisibili*, perché ciò consente di unire assieme linee visibili in un modello continuo di programmazione.

Le linee da 70 a 100 disegnano il contorno del castello. Alla fine di questo frammento di programma, sono stati letti 38 elementi di DATA. Ora si aggiungano queste linee e dopo il RUN compariranno le torri:

```
110 FOR K=1 TO 33
120 LET X=X-4
130 LINE-(X,Y),PRESET
140 IF Y=74 THEN LET Y=78 ELSE LET Y
   =74
150 LINE-(X,Y),PRESET
160 NEXT K
```

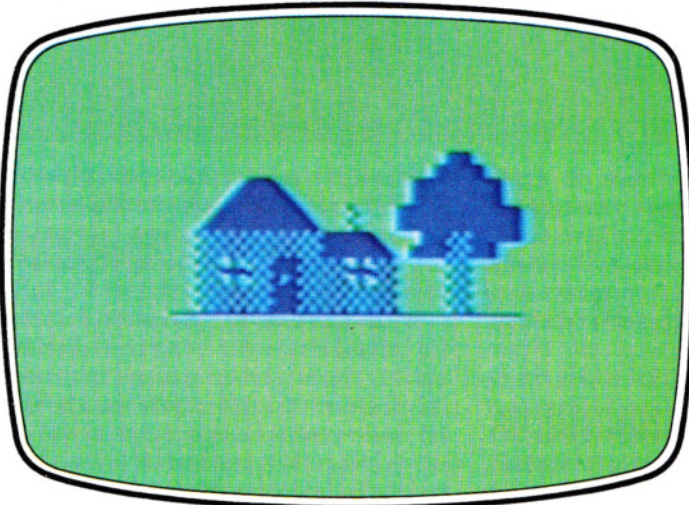
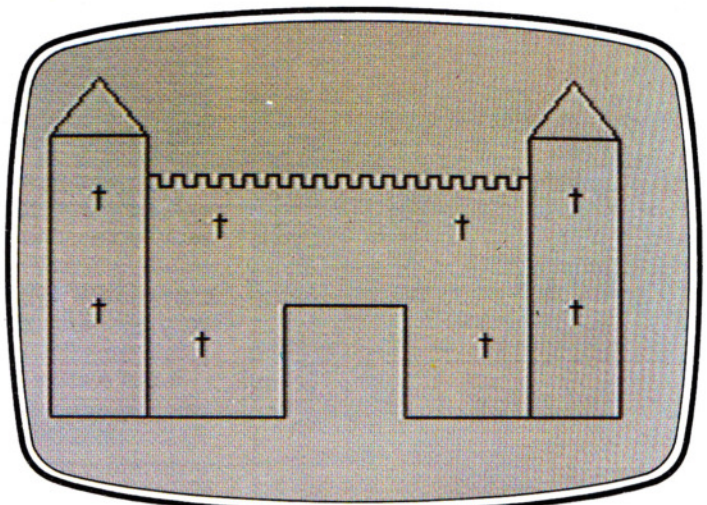
Forse ci si sta chiedendo che ne sia stato della linea READ. Questo è un caso in cui usare READ e DATA non fa risparmiare lavoro: occorrerebbero 66 voci di DATA per definire gli angoli delle torri, perciò il metodo qui usato è più conveniente, poiché ci risparmia di digitare una bella quantità di DATA. Queste linee disegnano qualche finestra:

```
170 FOR K=1 TO 8
180 READ X,Y
190 LINE(X,Y)-(X+4,Y),PRESET
200 LINE(X+2,Y-2)-(X+2,Y+6),
   PRESET
210 NEXT K
540 DATA 46,80,46,120,210,80,210,120,86,
   90,170,90,80,132,178,132
```

Si noti che non serve avere frasi DATA per ciascuna delle finestre, quattro gruppi in tutto: dato che sono tutte della stessa dimensione, basta un gruppo di DATA per finestra. Ora aggiungiamo queste linee e diamo il RUN per vedere cosa accade:

```
220 FOR K=1 TO 4000
230 NEXT K
240 CLS:PRINT@33,"PREMERE QUALSIASI
   TASTO PER CONTINUARE"
250 LET IN$=INKEY$:IF IN$=""
   THEN GOTO 250
270 GOTO 30
```

Premendo un tasto si otterrà OD, errore 'out of data'. La ragione è che il programma ha finito la lista di DATA. Per fortuna, non occorre immettere nuovamente tutte le DATA; basta aggiungere questa linea, che fa uso dell'istruzione RESTORE per far tornare il computer all'inizio della lista di DATA:

```
260 RESTORE
```

3. Esempio di grafica ROM sul Commodore

4. Un castello disegnato col Dragon/Tandy

# IMPARARE A CONTARE SU UN DITO

**I computer contano in base due: è come se avessero milioni di mani, ma con un solo dito ciascuna. Per programmare nel vero linguaggio della macchina, occorre imparare questo insolito sistema.**

Uno dei maggiori problemi, nell'apprendimento del codice macchina, consiste nella necessità di capire qualcosa sulla teoria dei numeri. Questo è un compito che non deve intimidire: se si sa contare fino a 16 non ci dovrebbero essere difficoltà. Ma, prima di tutto, occorre imparare a contare fino a due.

## PERCHÈ BASE DIECI?

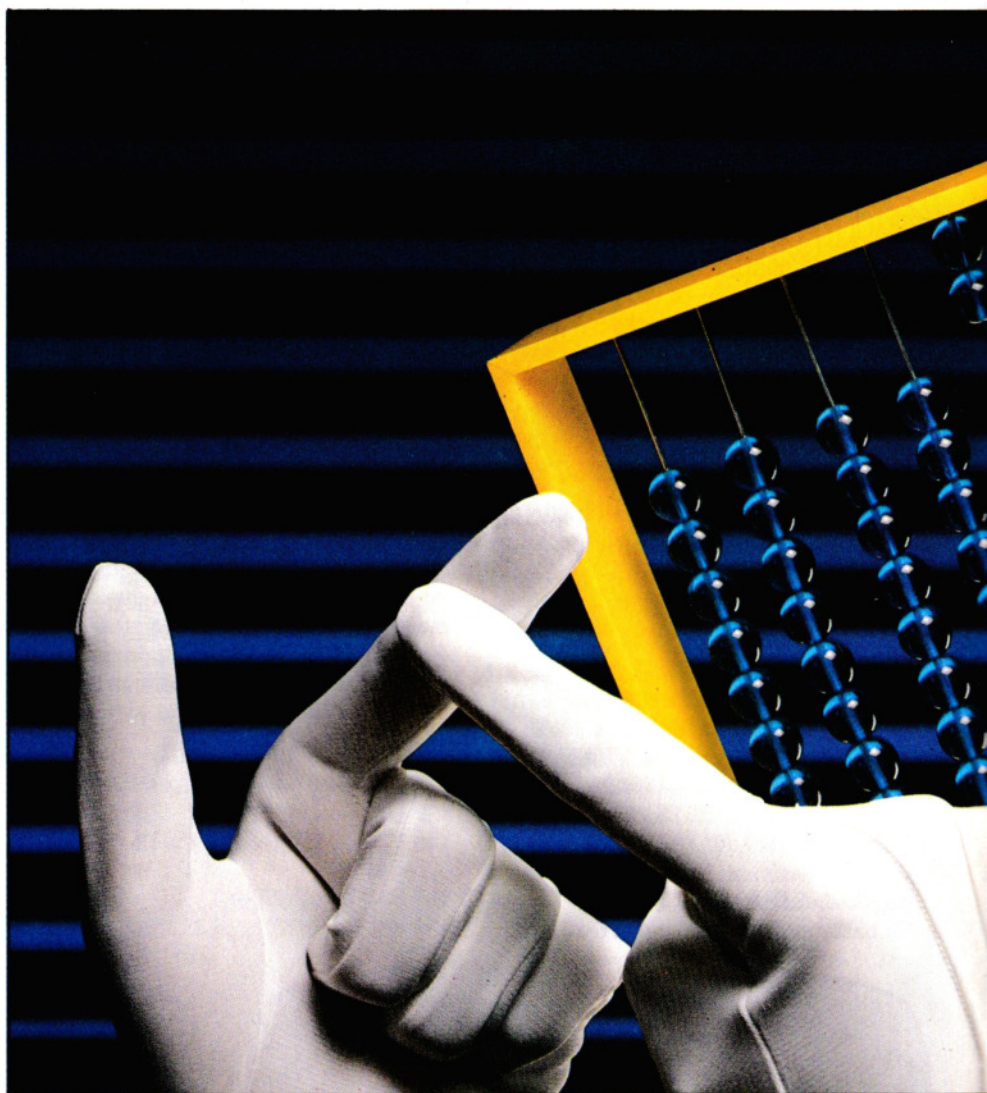
Anche la persona più 'antimatematica' non ha problemi a dire l'ora, a calcolare la durata d'un tempo in una partita o a sommare i punteggi in una classifica. L'uso dei numeri fa talmente parte della vita quotidiana, che non facciamo mai caso a come essi operano. Ma quando si inizia a programmare in codice macchina è necessario considerare più da vicino il modo in cui i numeri operano.

Nel mondo occidentale usiamo un sistema numerico basato sul numero 10. Ciò significa che per contare adoperiamo le cifre da 0 a 9. Quando ci serve un numero maggiore di 9, avendo usato ormai tutte le cifre disponibili, scriviamo un 1 in una colonnna a sinistra e uno 0 nel posto a destra. Dieci quindi è rappresentato da due delle cifre che abbiamo usato prima.

Se continuiamo a contare, scorriamo di nuovo le cifre da 0 a 9 nel posto di destra e quando abbiamo raggiunto 9 e ne aggiungiamo ancora una, il posto a sinistra scatta ancora di 1. E quando quel posto è stato riempito fino a 9, dobbiamo ancora aggiungere una colonna: compare un 1 nel posto a fianco a sinistra e i due posti a destra si riportano a 0. Questo viene chiamato sistema numerico in *base dieci*, o *decimale*, perché il valore della cifra è aumentato da un *moltiplicare* dieci in ciascun posto contando da destra. Per esempio, il numero 3.275 ha il valore di: $5+7\times10+2\times10\times10+3\times10\times10\times10$ oppure $5+70+200+3.000$.

Ciascuna cifra aumenta il suo valore di dieci volte il suo valore nominale per ogni posizione in cui essa viene spostata verso sinistra.

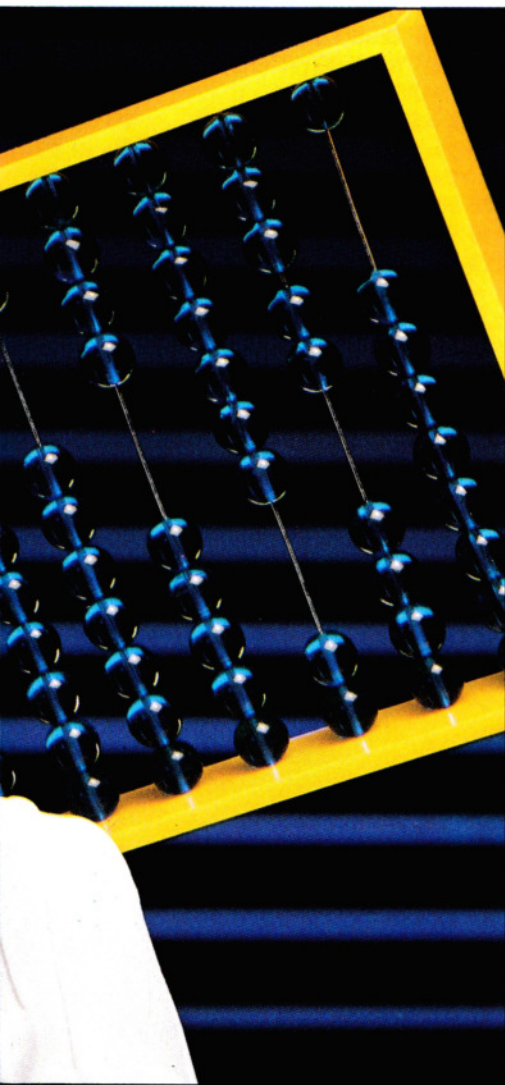


## ANTICO E MODERNO

Gli antichi Babilonesi avevano un sistema di numeri basato su una unità di misura di 60, che ancora oggi influisce su alcune unità di misura: il tempo e gli angoli. Ci sono 60 secondi in un minuto, 60 minuti in un'ora, un cerchio completo è sei volte 60 gradi, cioè 360 gradi, e ogni grado, a cui corrisponde il suo arco, misura 60 minuti primi.

Si può contare fino a 59 secondi, ma ne basta ancora uno e si passa a 1 minuto e si riparte da zero nel conto dei secondi. E quando si arriva a 59 minuti e 59 secondi, ancora un secondo e si avrà 1 ora, mentre minuti e secondi tornano a 0 prima di ripartire col conteggio.

Tracce di sistemi numerici con altre basi si ritrovano anche nel vecchio sistema anglosassone di misurazione (per esempio, ci sono 8 pinte in 1 gallone, oppure 12 pollici in 1 piede).

Si usava la base 12 nel sistema monetario britannico perché è facile da dividere per 2,3,4,6. I soldi devono spesso essere distribuiti tra più di una persona e la base 12 rende queste transazioni molto più facili da eseguire. Il numero 10, invece, è divisibile soltanto per due numeri, 2 e 5.

Spesso conviene usare un sistema numerico con una base diversa da dieci. (Per

**1. Le nostre dieci dita ci portano a contare per dieci e gli strumenti per contare come il semplice abaco o il pallottoliere sono modelli meccanici delle nostre dita**

esempio una libbra può essere riportata a once con successivi dimezzamenti.)

## NOVE MA BUONI

Non c'è ragione, in effetti, per non usare un sistema basato su qualsiasi numero si voglia. Per esempio, se tutti fossimo nati con un dito in meno in una mano, avremmo probabilmente un sistema *nonario* o in base nove. Sapremmo contare scorrendo le cifre da Ø a 8 nel modo normale. Ma se ne aggiungessimo ancora 1 a 8, scatterebbe la cifra 1 nel posto a fianco a sinistra e il primo posto tornerebbe Ø. In altre parole, il numero nove, in nonario, sarebbe rappresentato da 1Ø.

E ancora, se si aggiungesse 1 a 18 si otterebbe 2Ø e 1 più 88 darebbe 1ØØ.

Come si può vedere dall'esempio appena citato, le comuni regole dell'aritmetica si applicano qualunque sia il numero su cui si basa il sistema numerico. Per prova, si possono fare facili somme in un sistema a base otto o sette. Qualsiasi base si scelga per fare calcoli, si continuano ad applicare le comuni regole aritmetiche.

Nel nostro sistema nonario, per esempio, ogni movimento di una cifra di un posto a sinistra accresce il suo valore nominale secondo un moltiplicatore nove. Così, 3.275, in nonario, è uguale a $5+7\times9+2\times9\times9+3\times9\times9\times9$ o 2.417 in decimale. Ma, si noterà, non c'è la cifra 9 nel sistema nonario. Il nove è rappresentato da 1Ø. Sarebbe perciò più o meno corretto dire che 3.275 è uguale a $5+7\times1Ø+2\times1Ø\times1Ø+3\times1Ø\times1Ø\times1Ø$ in nonario.

Con un po' di agilità mentale, si dovrebbe essere capaci di fare piccole somme in nonario. Per esempio, $99\times9$ è uguale a 891 in decimale. In nonario 99 è rappresentato da 12Ø, cioè $1\times9\times9+2\times9$ o $81+18$, e 9 è 1Ø.

Perciò in nonario $99\times9$ si traduce in $12Ø\times1Ø$ cioè 1.2ØØ o $1\times9\times9\times9+2\times9\times9$ o 891 in decimale.

Ogni addizione, sottrazione, moltiplicazione o divisione di numeri nonari funzionerà. Attenzione, però; si ricordi, per esempio, che 16 in decimale è 17 in nonario: se si fa un riporto o un prestito in nonario, esso vale nove e, nella divisione, il 3 sta nel 1Ø tre volte.

## PROBLEMI DEGLI ULTRADECIMALI

Tuttavia, passare a sistemi numerici con basi maggiori di dieci è un po' difficile. Per maneggiarli con disinvoltura bisogna inventare nuove cifre. Per esempio, se si dovesse usare un sistema numerico in base dodici, si dovrebbero inventare cifre per rappresentare dieci e undici. In dodecimale, 1Ø significherebbe $1\times12$, o 12. E 11 sarebbe $1\times12+1$, cioè 13 in decimale.

Il modo più semplice di estendere il raggio delle cifre è usare l'alfabeto. Dieci potrebbe essere A e undici B. Alcuni numeri dodecimali sarebbero come questi: A2, BA e 7B, cioè 122, 142 e 95 in decimale.

**Un modello per numeri in basi diverse**

Può essere difficile visualizzare come funzionano sistemi numerici diversi, essendo abituati a contare per decine. In fondo, non è facile tagliarsi via qualche dito o farsene crescere in più, per poter contare su questi! Invece è possibile, ad esempio, unire due dita insieme per provare la base nove o prendere due bastoncini contrassegnati per contare in base dodici e così via. Ma un modo migliore è costruirsi un pallottoliere, simile a quelli mostrati in queste pagine. Proprio come un comune pallottoliere o abaco può aiutare a chiarire l'aritmetica in base 1Ø, così si può realizzare la struttura di un pallottoliere con tante palline su ogni colonna quante se ne desiderano. Non è necessario neppure creare una vera struttura: basta un sistema di scanalature in cui far rotolare avanti e indietro le palline. Non si dimentichi che in ogni fila ci vuole una pallina in meno del numero della base. Perciò in base dieci abbiamo nove palline e nel binario solo una.

Se non scorre dall'altra parte nessuna pallina è uguale a zero; se ne scorre una è uguale a uno ecc.

Quando si va oltre nove, se ne 'prende' una sulla fila accanto e si fanno scorrere tutte le palline a zero.

## UNA CALCOLATRICE IN TUTTE LE BASI

Fare i conti in basi diverse da dieci può essere faticoso; ecco quindi un programma che farà i conti in tutte le basi fino a 36. All'avvio, richiederà la base da usare. Il numero voluto va digitato in decimale.

Viene poi richiesto un numero: questo andrà introdotto nella base scelta. Se si è scelta la base 8, non si dovrebbero usare cifre oltre il 7, per esempio. E se si vuole una base maggiore di dieci, le cifre sopra il 9 dovranno essere digitate usando lettere dell'alfabeto, maiuscole. Dieci sarà A, undici B, dodici C e così via.

Poi il computer chiede un segno aritmetico (+, –, * o /), prima di richiedere un secondo numero nella medesima base.

Se si tratta di una divisione e il risultato non è un numero intero, il computer ci avvertirà con un messaggio "Non va bene" sullo schermo. Altrimenti, il risultato viene visualizzato nella base scelta.

```
20 INPUT "BASE (FINO A 36)";B%
30 INPUT "NUMERO (INTERO)";A$
40 INPUT "SEGNO";S$
50 INPUT "NUMERO (INTERO)";B$
60 PRINT ""
70 P$=A$:GOSUB 200
80 DA=D%
90 P$=B$:GOSUB 200:DB=D%
100 IF S$="*" THEN X%=DA*DB
102 IF S$="+" THEN X%=DA+DB
104 IF S$="–" THEN X%=DA–DB
106 IF S$="/" THEN X%=DA/DB
120 AN$=""
130 IF B%*(X%/B%–INT(X%/B%))>9
   THENAN$=CHR$(55+B%*(X%/B%–INT
   (X%/B%)))+AN$:GOTO150
140 AN$=RIGHT$(STR$(B%*(X%/B%–INT
   (X%/B%))),1)+AN$
150 X%=INT(X%/B%)
160 IF X%>0 THEN GOTO130
180 PRINT "BASE";B%":□"
   A$S$B$"="AN$
190 END
200 D%=0:IX%=0
205 Y=ASC(RIGHT$(P$,1))
210 IF Y<58 THEN D=Y–48: GOTO 230
220 D=Y–55
230 D%=D%+D*B%↑IX%:P$=LEFT$
   (P$, LEN(P$)–1):IX%=IX%+1
240 IF LEN(P$)>0 THEN GOTO 205
250 RETURN
```

Sullo ZX81, digitare il seguente programma, in lettere maiuscole, omettendo completamente la linea 10, omettendo LINE nelle linee 30, 40 e 50 e usando ** invece di ↑ nella linea 220.

```
10 POKE 23658,8
20 INPUT "BASE (FINO A 36)□";b
30 INPUT "NUMERO (INTERO)□";LINE a$
40 INPUT "SEGNO□"; LINE s$
50 INPUT "NUMERO (INTERO)□" LINE b$
60 CLS
70 LET p$=a$: GOSUB 180
80 LET x$=STR$ dec
90 LET p$=b$: GOSUB 180
95 LET y$=STR$ dec
100 LET z$=x$+s$+y$: LET x=VAL z$
110 IF x<>INT x THEN LET n$="NON
   VA BENE!": GOTO 160
120 LET n$=""
130 LET u=INT (x/b): LET u=u*b: IF x–u
   >9 THEN LET n$=CHR$ (55+(x–u))
   +n$: GOTO 140
135 LET n$=STR$ (x–u)+n$
140 LET x=INT (x/b)
150 IF x>0 THEN GOTO 130
160 PRINT AT 10,0;a$+s$+b$;"=□";n$
170 STOP
180 LET dec=0
190 LET index=0
200 LET y=CODE p$(LEN p$)
210 IF y<58 THEN LET d=y–48: GOTO
   220
215 LET d=y–55
220 LET dec=dec+d*b↑index
230 LET p$=p$(1 TO (LEN p$)–1)
240 LET index=index+1
250 IF LEN p$>0 THEN GOTO 200
260 RETURN
```

```
10 MODE6:PRINT"
20 INPUT "BASE (FINO A 36)",B%
30 INPUT "NUMERO (INTERO)",A$
40 INPUT "SEGNO",S$
50 INPUT "NUMERO (INTERO)", B$
60 CLS:PRINT"
70 P$=A$:GOSUB180
80 DECA$=STR$(DEC%)
90 P$=B$:GOSUB180: DECB$=STR$(DEC%)
100 X%=EVAL(DECA$+S$+DECB$)
110 IF X%<>EVAL(DECA$+S$+
   DECB$) THEN AN$="NON VA BENE!"
   : GOTO 160
120 AN$=""
130 IF X%MODB%>9THEN AN$=CHR$(55
   +X%MODB%)+AN$ ELSEAN$=STR$
   (X%MODB%)+AN$
140 X%=X%DIVB%
150 IFX%>0 GOTO 130
160 PRINTTAB(0,12) A$+S$+B$+"="+
   AN$
170 END
180 DEC%=0
190 INDEX%=0
200 LET Y=ASC(RIGHT$(P$,1))
210 IF Y<58 THEN D=Y–48 ELSE D=
   Y–55
220 DEC%=DEC%+D*B%∧INDEX%
230 P$=LEFT$(P$,LEN(P$)–1)
240 INDEX%=INDEX%+1
250 IF LEN(P$)>0 THEN GOTO 200
260 RETURN
```

```
10 CLS
20 INPUT"BASE (FINO A 36)";B
30 INPUT"NUMERO (INTERO)□";A$
40 INPUT"SEGNO□";S$
50 INPUT"NUMERO (INTERO)□";B$
60 P$=A$:GOSUB210
70 X$=STR$(DE)
80 P$=B$:GOSUB 210
90 Y$=STR$(DE)
100 IFS$="*" THENX=VAL(X$)*VAL(Y$)
110 IFS$="/" THENX=VAL(X$)/VAL(Y$)
120 IFS$="+" THEN X=VAL(X$)+VAL
```

**2. Ogni sistema numerico ha un suo 'modello' meccanico: così l'orologio è in base 60, il pallottoliere in base 10.**

```
    (Y$)
130 IFS$="−" THENX=VAL(X$)−VAL
    (Y$)
140 IFX<>INT(X) THENN$=
    "NON VA BENE!":GOTO190
150 U=B*INT(X/B):IF X−U>9 THEN N$=
    CHR$(55+(X−U))+N$:GOTO170
160 N$=MID$(STR$(X−U)+N$,2)
170 X=INT(X/B)
180 IFX>0 GOTO150
190 PRINT@257,A$+"□"+S$+"□"+B$;
    "=□";N$
200 END
210 DE=0
220 IN=0
230 Y=ASC(RIGHT$(P$,1))
240 IFY<58 THEN D=Y−48:GOTO260
250 D=Y−55
260 DE=DE+D*B↑IN
270 P$=LEFT$(P$,LEN(P$)−1)
280 IN=IN+1
290 IFLEN(P$)>0 THEN GOTO230
300 RETURN
```

Quando si comincia a far pratica con questi conti in basi differenti, ci si accorge che il computer bara. Infatti, questo programma, in realtà, converte i numeri immessi in decimali, fa i conti, poi riconverte il risultato nella base scelta. Diversamente dalla flessbile mente dell'uomo, il computer può fare i conti soltanto in decimale in un programma BASIC. (Questo, nonostante esso converta i numeri in binario quando traduce le istruzioni per eseguirle in linguaggio macchina!)

## ED ORA, BINARIO!

Nei computer, il sistema numerico più conveniente da usare è quello in base due. Questo perché un computer è composto di circuiti elettronici di commutazione che hanno due stati distinti: accesso (on) e spento (off).

Lo stato "off" rappresenta la cifra Ø e quello "on" la cifra 1. E queste, in base 2, sono tutte le cifre che servono.

Il sistema numerico in base 2 è noto come *binario*. È composto di nient'altro che Ø e 1: tutte le altre cifre sono state abolite. Perciò, se si parte a contare da Ø nel modo normale, si può arrivare fino a 1. Si aggiunga 1 a 1 e 1 scatta nel posto accanto a sinistra, mentre il primo posto torna sullo Ø.

Così in binario 1+1=Ø.

In binario, la numerazione da zero a otto dà: Ø, 1, 1Ø, 11, 1ØØ, 1Ø1, 11Ø, 111, 1ØØØ. Ancora una volta questi numeri obbediscono tutti alle comuni leggi aritmetiche. Per fare un esempio, se si somma 1Ø+11, prima si aggiungono le due cifre a destra, Ø+1; poi le due cifre a sinistra, 1+1 che dà come risultato 2, ma, come abbiamo appreso, due in binario è 1Ø. Così 1Ø+11=1Ø1: in altre parole, 2+3=5.

In binario, la sottrazione segue lo stesso semplice procedimento. La sola cosa cui prestare attenzione è il momento del riporto.

L'aspetto positivo del binario è che anche per gli esseri umani moltiplicazione e divisione sono straordinariamente facili. A ogni livello di una moltiplicazione, o si sta moltiplicando Ø per Ø (che dà come risultato Ø), o Ø per 1 (che dà ancora Ø), o 1 per 1 (che dà 1).

In binario, anzi, la moltiplicazione risulta talmente facile che la sa fare anche un computer!

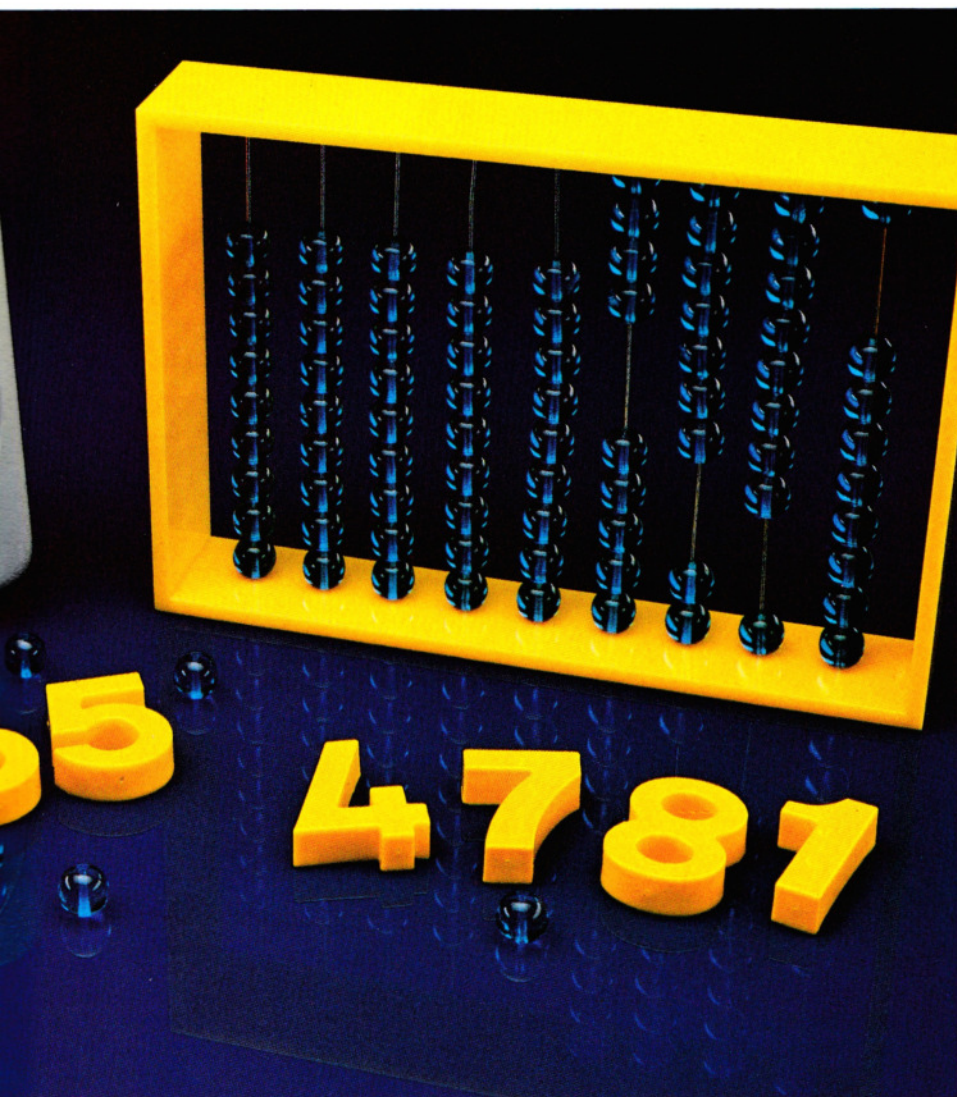
Per le divisioni è altrettanto semplice: ad ogni livello, si deve soltanto decidere se il divisore sta nel numero che si sta dividendo una o zero volte.

Si faccia qualche prova per vedere come tutti i procedimenti aritmetici funzionano in binario. Alla pagina seguente è riportata una serie di esempi di addizione, sottrazione, moltiplicazione e divisione già risolti.

## BIT E BYTE

I numeri binari rispecchiano fedelmente cosa succede nel computer. Ogni istruzione o frammento di DATA introdotti nel computer vengono tradotti in codice binario, manipolati e immagazzinati dai circuiti di commutazione.

Quando si comprende il binario, si comincia a comprendere il modo di operare del computer. Ogni cifra di un numero binario è rappresentata elettronicamente nel computer da un circuito acceso (on) o spento (off). Sull''on' il valore è 1. Sull''off' il valore è Ø. Ogni circuito a cifra binaria è noto come bit.

Per memorizzare e manipolare numeri binari, questi circuiti sono organizzati in

entità più ampie, capaci di rappresentare numeri più grandi e più utili. In quasi tutti gli home computer, i bit sono organizzati in gruppi di otto, formando il cosiddetto “byte”, di otto bit.

Ciò vuol dire che ciascun byte può rappresentare otto cifre binarie, ossia contenere tutti i numeri tra ØØØØØØØØ, o zero, e 11111111, o 128+64+32+16+8+4+2+1, cioè 255.

Numeri maggiori di 255 sono rappresentati da due o più byte.

## FRAZIONI BINARIE

Tutti i programmi e gli esempi mostrati finora hanno riguardato numeri interi, ma è possibile convertire anche frazioni in numeri binari.

Per esempio, 1/2 è Ø,1 in binario, 1/4 è Ø,Ø1, 1/8 è Ø,ØØ1. Si può già capire il modello che si sta delineando. Di fatto, ogni frazione può essere costituita da 1/2, 1/4, 1/8, 1/16 ecc. È indubbiamente più difficile calcolare l'equivalente binario di frazioni perché la serie di Ø e 1 spesso risulta infinita!

Nel convertire frazioni, si può pensare a 1/2, 1/4 ecc. come Ø,5 Ø,25 e così via. Si può allora calcolare facilmente che una frazione come Ø,75 è costituita da 1*Ø,5+1*Ø,25, il che vuol dire che il suo equivalente binario è Ø,11.

Il seguente programma provvede a queste conversioni al computer:

```
10 CLS
20 PRINT@35,"CONVERSIONE DA DECIMALE
   A BINARIO"
30 PRINT: INPUT"IMMETTERE UN NUMERO
   TRA Ø E 1";N
40 IF N<=Ø OR N>=1 THEN 30
50 N$="Ø."
60 FOR T=1 TO 30
70 N=N*2
80 N$=N$+CHR$(48+INT(N))
90 N=N-INT(N)
100 NEXT T
110 PRINT@257,"EQUIVALENTE IN
   BINARIO:"
120 PRINT: PRINTN$: PRINT
130 PRINT"ANCORA UN NUMERO (S/N)?"
140 A$=INKEY$
150 IF A$="S" THEN 10
160 IF A$<>"N" THEN 140
170 END
```

```
20 PRINT "♥CONVERSIONE DA DECIMALE
   A BINARIO"
30 PRINT "▣IMMETTERE UN NUMERO TRA
   Ø E 1";:INPUT N
40 IF N<=Ø OR N>=1 THEN 20
50 N$="Ø."
60 FOR T=1 TO 32
70 N=N*2
80 N$=N$+CHR$(48+INT(N))
90 N=N-INT(N)
100 NEXT T
110 PRINT "▣EQUIVALENTE IN BINARIO:"
120 PRINT "▣"N$
130 PRINT "▣ANCORA UN NUMERO
   (S/N)?"
140 GET A$
150 IF A$="S" THEN RUN
160 IF A$<>"N" THEN 140
170 PRINT "♥": END
```

```
10 CLS
20 PRINT '"CONVERSIONE DA DECIMALE A
   BINARIO"
30 INPUT "IMMETTERE UN NUMERO TRA Ø
   E 1",N
40 IF N<=Ø OR N>=1 THEN GOTO 30
45 PRINT ''"DECIMALE:□";N
50 LET N$="Ø."
60 FOR T=1 TO 16
70 LET N=N*2+1E-9
80 LET N$=N$+CHR$ (48+INT N)
90 LET N=N-INT N
100 NEXT T
110 PRINT ''"EQUIVALENTE IN BINARIO:"
120 PRINT N$
130 PRINT ''''"ANCORA UN NUMERO
   (S/N)?"
140 LET A$=INKEY$
150 IF A$="S" THEN GOTO 10
160 IF A$<>"N" THEN GOTO 140
170 STOP
```

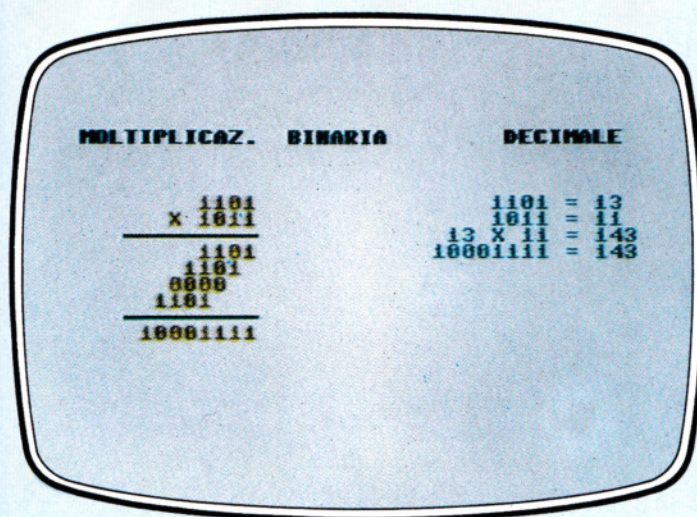
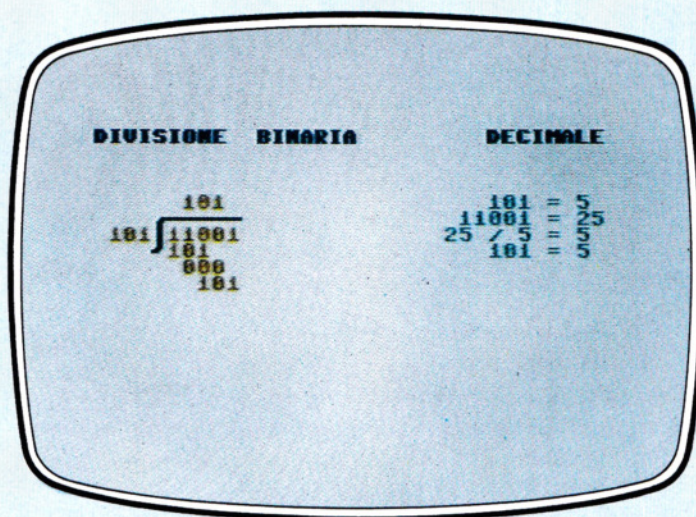
```
10 MODE 1
20 PRINT'"CONVERSIONE DA DECIMALE A
   BINARIO"
30 INPUT''"IMMETTERE UN NUMERO TRA Ø
   E 1",N
40 IF N<=Ø OR N>=1 THEN 30
50 N$="Ø."
60 FO T=1 TO 32
```

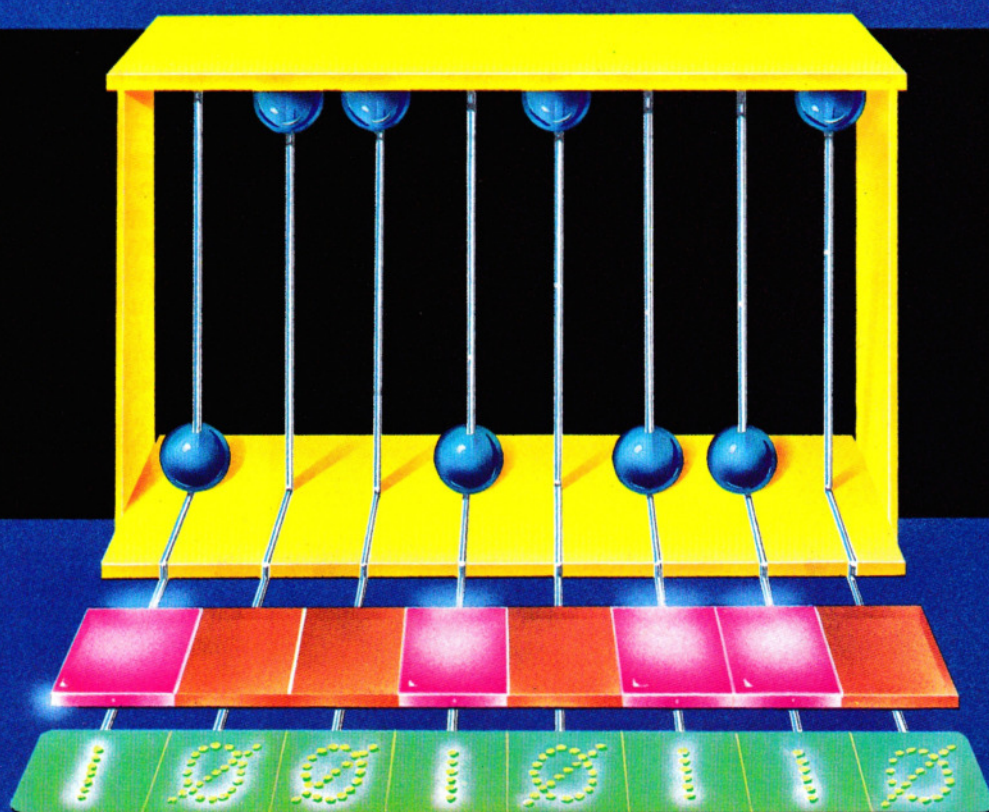
**1. L'addizione binaria opera da destra a sinistra: Ø+Ø dà Ø, Ø+1 dà 1, 1+1 dà 1Ø, cioè Ø col riporto di 1. Controlliamo l'operazione nella somma qui riportata**

**2. La sottrazione binaria è ugualmente semplice: 1−1 dà Ø, 1−Ø dà 1, mentre, se si vuole sottrarre 1 a Ø, si deve riportare questo nella cifra a fianco a sinistra**

```
70 N=N*2
80 N$=N$+CHR$(48+INT(N))
90 N=N-INT(N)
100 NEXT
110 PRINT''"EQUIVALENTE IN BINARIO:"
120 PRINTN$
130 PRINT''''"ANCORA UN NUMERO (S/N)?"
140 A$=GET$
150 IF A$="S" THEN 10
160 IF A$<>"N" THEN 140
170 END
```

Il programma accetta un qualsiasi numero tra Ø e 1 e ne stampa l'equivalente binario. La linea 7Ø parte raddoppiando il numero digitato, poi la linea 8Ø costruisce il numero binario, un carattere per volta. Il computer calcola per primo il valore di INT (N), che sarà o Ø o 1, poi lo somma a 48 per ottenere 48 o 49: (i codici ASCII di "Ø" e "1"). Ciò che in effetti si sta facendo è tramutare i numeri in stringhe, in modo da poterli raccogliere in N$. La linea 9 toglie da N l'intero per lasciare solo la parte frazionaria. I programma dello Spectrum è leggermente diverso perché, mentre INT 1 è 1, come ci si aspetterebbe, INT (Ø,5*2) risulta Ø. Questo perché Ø,5*2 è immagazzinato come Ø,99999999... Perciò, bisogna aggiungere una piccola frazione a N alla linea 7Ø, per bilanciare e far funzionare INT come dovrebbe.

**3. Un abaco binario richiede solo singoli contatori, mentre in un computer questo compito è svolto da impulsi di elettricità in un circuito**

**3. Ad ogni passaggio di una moltiplicazione binaria, o si moltiplica Ø per Ø, che dà Ø, o Ø per 1, che dà ancora Ø, o 1 per 1 che dà 1. Seguiamo l'esempio riportato qui sopra**

**4. In una lunga divisione binaria, si deve verificare se il divisore entra nel dividendo una volta, nel qual caso si segna 1, o nessuna volta, e allora si segna Ø**

# DALL'ABACO ALL'ELETTRONICA

I numeri binari possono essere sommati, sottratti, moltiplicati e divisi come qualsiasi altro numero, ma vengono scelti per essere usati nei computer per le loro particolarità: è infatti facile verificare se un circuito è acceso (on) o spento (off).

Le cifre binarie sono rappresentate dai circuiti del computer. Quando un circuito è off, esso rapresenta uno Ø, quando è on rappresenta un 1. Nei chip degli home computer, i circuiti binari sono ordinati per otto, così possono essere rappresentati numeri binari di otto bit. Questo offre un campo da ØØØØØØØØ, o Ø in decimale, fino a 11111111, o 255 in decimale. Se si dà una scorsa al manuale di un computer, si scopre che il numero 255 (o 256 se si parte a contare da 1 invece che da Ø) ricorre più volte.

Per alcune applicazioni sono necessari numeri più grandi e si abbinano due locazioni di memoria binaria di otto bit per rappresentare un numero binario di 16 bit. Il microprocessore Dragon può operare con questi e talvolta conta in binario con 16 bit.

In questo caso, è possibile operare con numeri tra ØØØØØØØØØØØØØØØØ e 1111111111111111, ossia con numeri tra Ø e 65535 in decimale.

# SCRITTURA SU SCHERMO

- USO DI **TAB** CON **PRINT** E **INPUT**
- POSIZIONARE CON LA PUNTEGGIATURA
- UN DISPLAY ORDINATO
- COME USARE LE SPAZIATURE

**Una buona visualizzazione sullo schermo rende leggibili le istruzioni e comprensibili le tabelle. Si tratta di usare i comandi giusti.**

PRINT e INPUT sono i primi due comandi imparati dai programmatori. Si usano tutte le volte e in tutti i tipi di programma ed è necessario avere un'idea precisa del loro funzionamento: può darsi che non li stiamo sfruttando al massimo.

Ci sono molti modi di redigere un testo sullo schermo e una buona 'presentazione' o display è importante, specialmente se si scrivono programmi che poi altri useranno. Se lo schermo è zeppo di istruzioni confuse o il testo è stampato a casaccio, questo può scoraggiare molti ad applicarsi alla sua lettura. E così quiz, grafici o giochi, realizzati con tanta fatica, resteranno inutilizzati.

Perciò, qualsiasi programma si stia scrivendo, il testo e le istruzioni devono essere facili da leggere: se si vuole che l'operatore risponda correttamente a una richiesta di INPUT, deve essere chiaro cosa gli si chiede.

Ogni computer ha il suo modo di posizionare i testi sullo schermo. Generalmente, i comandi sono TAB, AT, il simbolo @ o i caratteri per il controllo del cursore. Per ottenere formati diversi di visualizzazione, si usa anche la punteggiatura: virgole, punti e virgole e apostrofi. Questa lezione spiega come fare buon uso di tali comandi per ottenere display ordinati e leggibili, di pronta comprensione.

Si sa già che un comando come:

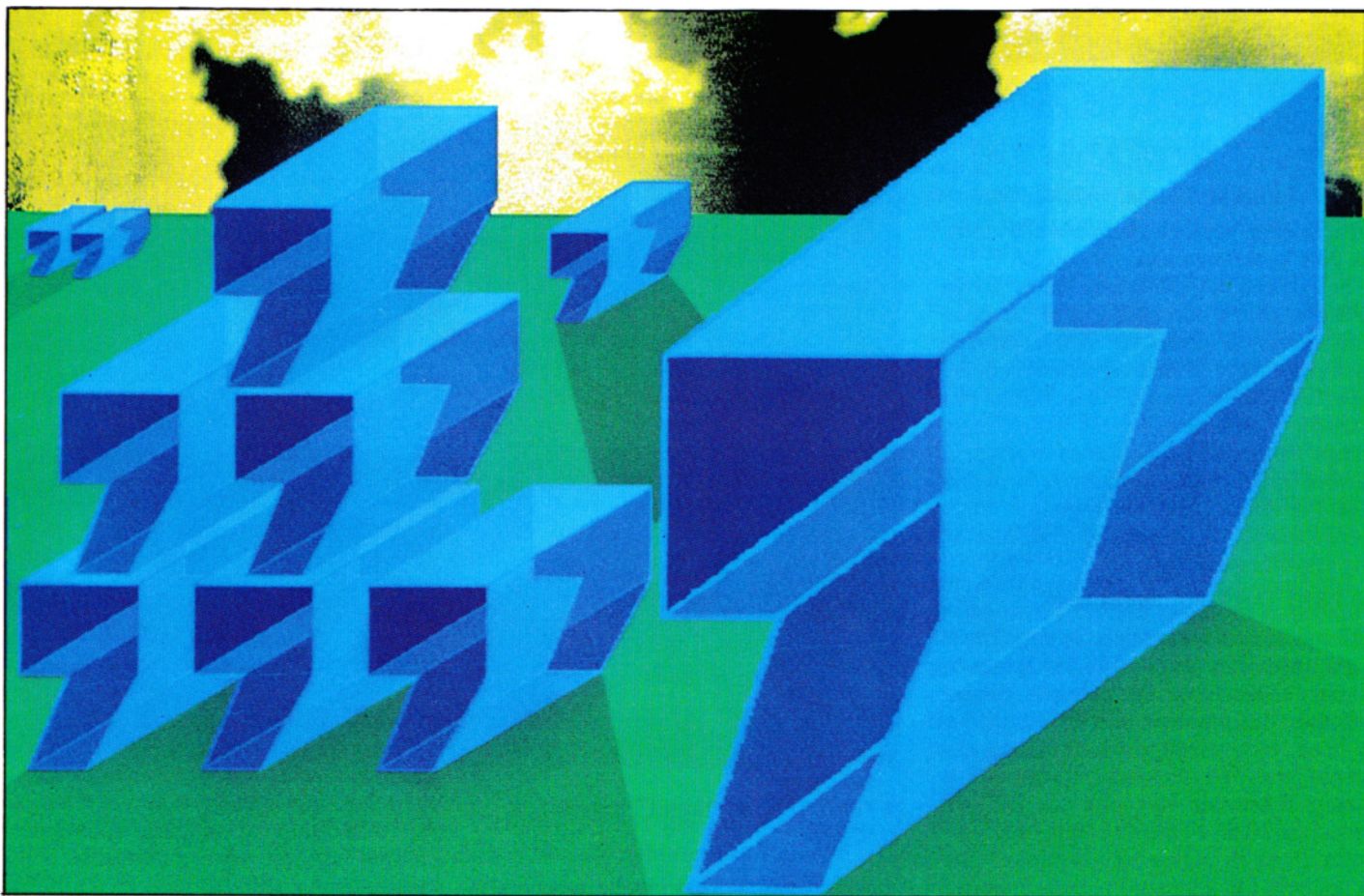
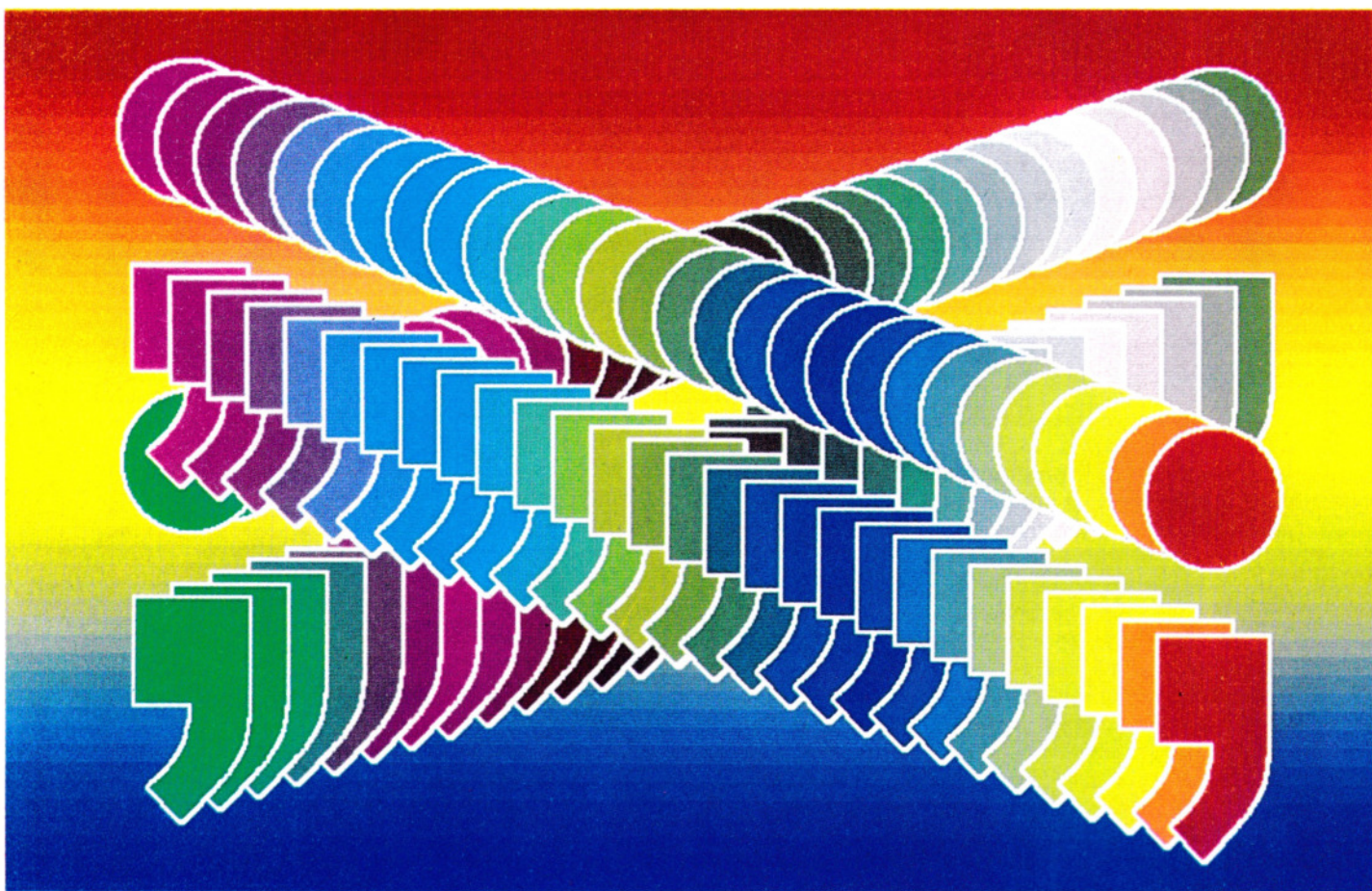
```
PRINT "Buon giorno"
```

visualizza il messaggio sulla prima linea libera accostandolo a sinistra. Se prima si pulisce lo schermo:

```
10 CLS
20 PRINT "Buon giorno"
```

il messaggio appare sulla prima linea in alto dello schermo. Il computer si comporta come chiunque di fronte a un foglio bianco. Ma si può anche dire al computer di stampare il messaggio in una posizione diversa. Si aggiunga questa linea all'ultimo programma:

```
30 PRINT TAB(20) "Buona sera"
```

Le parole vengono adesso visualizzate a partire dalla colonna 20, in mezzo allo schermo. Volendo, ogni parola può venire

posizionata separatamente, usando due istruzioni TAB.

Proviamo a cambiare la linea 3Ø in:

```
30 PRINT TAB(2Ø) "Buona" TAB(3Ø) "sera"
```

Ora la prima parola è stampata alla colonna 2Ø e la seconda alla colonna 3Ø.

Si noti che *non* c'è spazio tra la parola TAB e la prima parentesi. Se si lasciasse uno spazio, il computer non capirebbe quel che si vuole e riporterebbe errore del tipo "no such variabile" (non esiste tale variabile). Se si desidera che il messaggio venga riprodotto su una linea diversa, si devono indicare due numeri tra le parentesi. Per esempio, la linea seguente visualizza come prima le parole alla colonna 2Ø, ma stavolta sulla linea 15:

```
40 PRINT TAB(2Ø,15) "Buona notte"
```

TAB con un solo numero è utile dopo che si è già usato un comando TAB per spostarsi su una linea diversa. Per esempio, per posizionare due parole (o numeri) su una sola linea, ci si porti prima sul primo elemento con TAB (colonna, linea) poi si stampi il secondo elemento con TAB (colonna): la PRINT è eseguita sulla linea corretta. Però, in questo caso, il numero nella seconda TAB è il numero di colonna dall'ultima riga indicata, *non* la posizione in assoluto sullo schermo. Questa, quindi, è una linea di programma accettabile:

```
50 PRINT TAB(2Ø,17) "primo" TAB(8)
   "secondo"
```

La parola *secondo* è riprodotta 8 colonne dopo *primo*, alla colonna 28, non alla 8 come si potrebbe pensare. Se si chiede al computer di scrivere oltre la fine della linea, con una TAB(85) per esempio, esso continua a contare sulla linea successiva, o anche su diverse linee ulteriori, finché non ha scorso il numero giusto di colonne.

Vale la pena di fare esperienze con il comando TAB per farsi un quadro esatto delle sue possibilità.

## UNA INPUT BEN DISPOSTA

TAB si può usare con INPUT come pure con PRINT. Ecco un esempio:

```
10 INPUT TAB(1,5)"Immetti il tuo
   nome□",nome$ TAB (1,6)"e quanti anni
   hai□",anni
20 PRINT TAB(1,Ø) "Nome:□";nome$;
   "□anni□";anni
```

Questo visualizza la prima istruzione alla colonna 1, linea 5 e poi chiederà con un interrogativo un elemento per la INPUT. Qualsiasi cosa si batta, viene assegnata a 'nome$'. L'istruzione successiva è riprodotta sulla linea successiva, esattamente sotto la prima, con un altro interrogativo per la seconda INPUT da inserire in età$. Questa è una variabile numerica, così va digitato un numero. La linea 2Ø visualizza semplicemente nome ed età per mostrare che l'informazione è di fatto immagazzinata. Se non si vuole il punto interrogativo, si tralascino semplicemente le virgole alla linea 1Ø. Ciò sarebbe indispensabile in una linea come:

```
INPUT "Prezzo dell'articolo:" prezzo
```

poiché un interrogativo disturberebbe: "Prezzo dell'articolo: 34ØØØ" è ovviamente meglio di "Prezzo dell'articolo:?34ØØØ".

Ecco un breve programma che fa uso di INPUT TAB: consente di immettere il numero di goal "in casa", "fuori casa" e il numero totale delle partite, calcolando una specie di media. Automaticamente, tutto viene ben incolonnato sullo schermo. Il valore calcolato dal computer appare nell'ultima colonna, immediatamente dopo l'immissione degli altri valori.

```
10 CLS
20 PRINT' TAB(1) "SQUADRA" TAB(13) "IN
   CASA" TAB(22) "FUORI" TAB(29) "PART."
   TAB(35) "MEDIA"
30 LET RIGA=3
40 INPUT TAB(1,RIGA) N$
50 INPUT TAB(13,RIGA) I
60 INPUT TAB(22,RIGA) R
70 INPUT TAB(30,RIGA) P
80 IF P<>0 THEN PRINT TAB(35,RIGA);
   INT((I+R)/P*100)/100 ELSE PRINT TAB
   (35,RIGA)"*"
90 RIGA=RIGA+1
100 GOTO 40
```

La media di goal per partita è calcolata alla linea 80: sembra un calcolo complicato, ma è il solo modo con cui il risultato viene dato in due decimali. La media pura e semplice è data dalla somma dei goal (in casa e fuori), diviso per il totale delle partite giocate, ma essa verrebbe calcolata fino a 9 decimali. È una pignoleria eccessiva per le nostre esigenze e guasterebbe anche la visualizzazione, dato che i numeri si sovrapporrebbero gli uni agli altri.

## L'USO DELLA PUNTEGGIATURA

Finora si è usata la TAB per posizionare testi o per collocare INPUT in vari punti dello schermo. Ma ci sono altri modi di disporre o visualizzare informazioni senza usare TAB. Basta usare la punteggiatura corretta, o virgole, o punti e virgola, o apostrofi. Quando si sanno padroneggiare questi 'trucchi', si scopre che in molti casi, specialmente con colonne di numeri o parole, l'informazione può venir posizionata facilmente senza troppa fatica e solo occasionalmente ricorrendo alla TAB.

Si provino queste poche linee per farsene un'idea:

```
5 CLS
10 PRINT "01234567890123456789012345
   67890123456789"
20 PRINT 9;9;9
30 PRINT 9,9,9
40 PRINT 9'9'9
```

(La linea 10 numera le colonne in cima allo schermo per dare un riferimento). Si faccia attenzione a come la punteggiatura determini dove i numeri sono stampati. I numeri separati da punto e virgola sono stampati uno accanto all'altro. Una virgola separa i numeri dentro 'campi', ciascuno di 10 colonne, mentre un apostrofo fa disporre ogni numero su una nuova linea.

Ora proviamo lo stesso programma con lettere (cioè stringhe) invece che numeri. Aggiungiamo queste linee prima di ridare il RUN:

```
45 LET A$="A"
50 PRINT A$;A$;A$
60 PRINT A$,A$,A$
70 PRINT A$'A$'A$
```

1. Il punto e virgola può essere usato con PRINT o INPUT per controllare la visualizzazione di testi. Ogni elemento è collocato sullo schermo subito dopo l'altro, sulla stessa linea

Si sarà notato che qui la punteggiatura si comporta ugualmente, ma le stringhe sono sempre stampate sulla sinistra di ciascun campo di 10 colonne, mentre i numeri sulla destra.

E ciò infatti ha un senso: se si scrivesse una colonna di parole, questa verrebbe di norma allineata a sinistra, ma una colonna di numeri quasi certamente verrebbe allineata a destra, in modo da avere bene in linea unità, decine, centinaia ecc. E così si comporta il computer.

Trascriviamo e avviamo il seguente programma (si osservi come il computer allinea e somma i numeri, come faremmo noi su un foglio di carta):

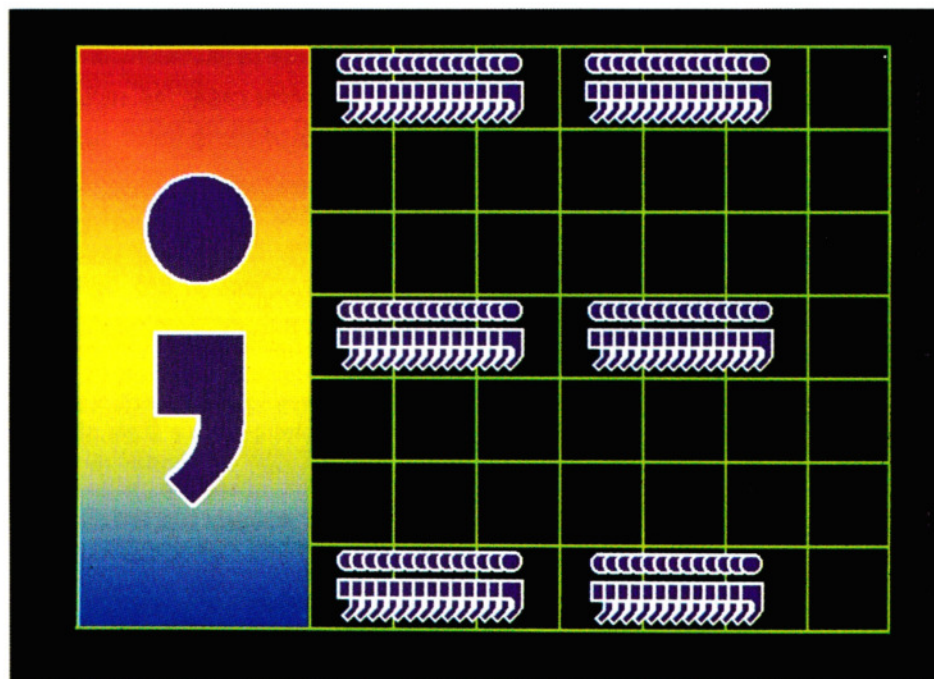
```
10 PRINT 14'67'245'42'811
20 PRINT '14+67+245+42+811
```

Se si vogliono diverse colonne, si usi la virgola. Ecco un programma per stampare il quadrato, il cubo e la quarta potenza di tutti i numeri da 1 a 10:

```
10 CLS
20 PRINT' TAB(4)"NUMERO" TAB(14) "AL
   QUADRATO" TAB(26)"AL CUBO" TAB(31)
   "ALLA QUARTA"
30 FOR J=1 TO 10
40 PRINT J,J*J,J*J*J,J*J*J*J
50 NEXT J
```

I numeri vengono stampati alla linea 40 e le virgole fanno sì che ciascun numero sia allineato in campi separati. Non occorre posizionare le intestazioni delle colonne con la TAB. Se qui si usassero le virgole, le intesazioni verrebbero allineate alla sinistra di ogni campo e non starebbero sopra i numeri.

Anche il punto e virgola è molto utile. Si usa spesso dopo un'istruzione TAB per stampare quello che viene in seguito senza interporre spazi. Il punto e virgola non è in pratica necessario nella stampa di stringhe perché, come ormai si sa, le stringhe sono sempre e comunque stampate sulla sinistra; invece è essenziale quando si tratta di numeri: in questo caso va adoperato, se si vuole che i numeri siano stampati secondo la TAB. Il punto e virgola è utile anche al termine di una PRINT, poiché mantiene il cursore dov'è, pronto per stampare l'elemento successivo, che può arrivare anche più tardi. Questo programma stampa l'alfabeto usando un ciclo FOR ... NEXT, che scorre i codici ASCII per i caratteri da A a Z:

```
10 CLS
20 FOR codice=65 TO 90
30 PRINT CHR$(codice);
40 NEXT codice
```

Lo si riprovi senza punto e virgola per verificare come ogni lettera venga stampata su una linea separata.

Segni di interpunzione e TAB offrono una tale varietà di modi di disporre testi sullo schermo, che si può star sicuri di trovare la combinazione giusta per ogni programma. Con un po' di pratica, si riesce a imparare esattamente qual è il ruolo di ogni separatore usato.

I computer Sinclair hanno tre istruzioni per dare aspetto ordinato alla scrittura su schermo: PRINT, PRINT AT e PRINT TAB.

PRINT, preceduta da un numero di linee e seguita da niente, è la più semplice. Così com'è, stampa una linea vuota. Un paio di esempi sono nel programma per la media dei goal (vedi sotto). PRINT AT, già incontrata, richiede una spiegazione più estesa. Lo schermo dello Spectrum ha 22 linee, numerate dall'alto da Ø a 21 e 32 colonne, o "spazi carattere", numerati (da sinistra) da Ø a 31. Il numero della linea si dà sempre per primo. Perciò queste linee:

```
PRINT AT Ø,Ø;"*"
PRINT AT 21,Ø; "*"
PRINT AT Ø,31; "*"
PRINT AT 21, 31; "*"
```

visualizzano una stella a ogni angolo dello schermo. Se si chiede al computer di riprodurre più di un carattere in una particolare locazione dello schermo, il primo è visualizzato dove richiesto, gli altri nelle posizioni successive, sulla stessa linea.

```
PRINT AT 1Ø, 12; "LUNGHEZZA"
```

... per esempio, dispone la 'L' in 1Ø, 12 e le altre lettere in 1Ø, 13 fino a 1Ø, 17 rispettivamente. Se ciò che si vuole scrivere non entra sulla linea, semplicemente deborderà nella linea sottostante, a volte alterandone il significato. Proviamo, ad esempio, con:

```
PRINT AT 5,29; "BARBARO"
```

Questa scrittura su schermo (l'esempio è stato prodotto su uno Spectrum) mostra come si possa usare un punto e virgola per unire messaggi. Si provi un programma come:

```
10 PRINT "messaggio";
20 GOTO 10
```

**2. Gli apostrofi, che non vengono usati su tutte le macchine, spesso sono utilizzati per istruire il computer a collocare il successivo elemento da visualizzare su una nuova linea (se ciò non avviene automaticamente)**

## PRINT TAB

L'istruzione PRINT TAB sullo Spectrum agisce per lo più come PRINT AT. Ci sono però due differenze:

1. Non importa specificare la linea.
2. Non si può usare PRINT TAB per sovrapporre, e quindi cancellare, qualsiasi cosa sullo schermo. Inserendo una nuova PRINT TAB sulla vecchia posizione, la stampa 'cala' di una linea. Per rendere chiara la differenza, prima si provi con:

```
PRINT AT Ø, 15; "*"
PRINT AT Ø, 15; "?"
```

Poi, ripulito lo schermo (CLS), con:

```
PRINT TAB 15; "*";
PRINT TAB 15; "?";
```

Per le ragioni esposte sotto, *si faccia molta attenzione alla punteggiatura.*

PRINT TAB non presenta grandi vantaggi rispetto a PRINT AT, a meno di non avere molto materiale da inserire in tabelle. Allora può far risparmiare molto tempo di battitura e la necessità di ricordare i numeri di linea a cui andare. Questo piccolo programma, per esempio, permette di immettere il numero di goal fatti "in casa" e "fuori", nonché il numero totale di partite giocate, facendone un tabulato ordinato sullo schermo. Viene calcolata anche la media gol/partita. Sullo ZX81, si immettano tutte le linee in lettere maiuscole.

```
1Ø PRINT TAB 14;"IN CASA";TAB 19;"FUORI";
   TAB 24;"PART.";TAB 29;"MEDIA"
15 PRINT
2Ø INPUT "Squadra?";n$
3Ø PRINT TAB Ø;n$;
4Ø INPUT "In casa?";i
5Ø INPUT "Fuori?";r
6Ø INPUT "Partite?";p
8Ø PRINT TAB 14;i;TAB 19;r;TAB 24;n;TAB
   29;INT ((I + R)/P)
9Ø PRINT
1ØØ PAUSE
11Ø GOTO 2Ø
```

Il programma ha un aspetto ordinato, sia grazie alla tabulazione, sia perché la linea 9Ø stampa una linea vuota sotto ogni immissione di dati.

È anche facile da usare: quando si dà per la prima volta il RUN, il programma chiede il nome della squadra, poi gli altri valori. Non occorre calcolare la media di goal per partita: ci pensa il computer.

Ci sono però due ostacoli nel programma, così com'è. Il primo è che la media calcolata dall'ultima parte della linea 8Ø è

sempre arrotondata per difetto in numero intero. Come affrontare queste complicazioni sarà detto più avanti; in ogni caso, se questa media ci sembrasse troppo grossolana, si potrebbe sempre introdurre una PRINT TAB a parte per la media, ottenendone il calcolo fino a dieci decimali!

L'altro intoppo è che, se immettessimo un numero di partite pari a zero, il programma si fermerebbe con un errore del tipo "Number too big", ossia numero troppo grande. Infatti, avremmo una divisione per zero nella linea 8Ø. Per ovviare a questo, cancelliamo la linea 8Ø attuale e aggiungiamo queste linee:

```
7Ø IF i=n THEN PRINT TAB 14;i;TAB
   19;r;TAB 24;n;TAB 29;"*"
8Ø IF i<>n THEN PRINT TAB 14;i;TAB
   19;r;TAB 24;n;TAB 29;INT (r/(i−n))
```

Per uscire dal programma senza ripulire lo schermo, è necessario premere CAPS SHIFT e BREAK (su ZX81, BREAK da solo). La linea 1ØØ dà il tempo di fare questa operazione prima della richiesta del nome seguente. Un programma per tabulare e calcolare nello stesso tempo è anch'esso semplice. Eccone un esempio elementare, i primi passi di un programma per un bilancio familiare:

```
1Ø PRINT TAB Ø;"DATA";TAB 1Ø;"ART.";TAB
   26;"PREZZO"
15 PRINT: PRINT
2Ø LET total=Ø
3Ø FOR I=1 TO 8
4Ø INPUT "Data?";d$
5Ø INPUT "Art.?";i$
6Ø INPUT "Prezzo?";c
7Ø PRINT TAB Ø;d$;TAB 1Ø;i$;TAB 26;c
75 PRINT
8Ø LET total=total+c
9Ø NEXT I
11Ø PRINT TAB 26;"– – – – – – –";
12Ø PRINT TAB 1Ø;"Total"; TAB 26;total
```

## ATTENZIONE AI PUNTINI

Tutti gli sforzi per ottenere una scrittura sullo schermo ordinata e pulita andranno a vuoto, se si sbaglia punteggiatura, specie in un'istruzione PRINT TAB.

Di norma, dopo un'istruzione PRINT TAB, si usano due punti e virgole. Per capire perché si provi questo programmino:

```
1Ø FOR n=1 TO 21
2Ø FOR t=Ø TO 24 STEP 6
3Ø PRINT TAB t; n;
4Ø NEXT t
5Ø NEXT n
```

Dopo il RUN, si cambi la linea 3Ø in:

```
3Ø PRINT TAB t, n;
```

e poi si confronti con questa:

Un altro esempio su schermo reale mostra la capacità dell'apostrofo in una linea di programma come:

```
1Ø PRINT "messaggio"'
2Ø GOTO 1Ø
```

```
3Ø PRINT TAB t; n
```

Può lasciare perplessi perché si ottengano questi risultati, ma non importa. Quel che conta è ricordare queste regole:

**1.** Un punto e virgola significa "di seguito (senza spazio di divisione) alla parola (o quello che sia) successiva"!

**2.** Una virgola significa "la parola successiva (o quello che sia) deve cominciare all'inizio della colonna, Ø o 15", che viene prima!

**3.** Un'istruzione PRINT TAB richiede *due* punti e virgola, oppure sarà un guaio.

Il modo più facile di scrivere un messaggio sullo schermo è:

```
PRINT "CIAO"
```

"CIAO" apparirà sullo schermo allineato a sinistra sulla prima linea a disposizione; se PRINT "CIAO" è sull'ultima linea dello schermo, viene eseguito uno *scrolling* per far spazio al messaggio. Ricordarsi di battere le virgolette! È in questa forma che rimangono sullo schermo la stesura del programma, come pure il messaggio. Meglio se il messaggio apparisse su uno schermo ripulito. Ecco cosa serve:

```
1Ø CLS
2Ø PRINT "CIAO"
```

Ciò equivale a dare al computer un foglio bianco per scrivere. "CIAO" compare adesso in alto a sinistra su uno schermo totalmente vuoto. Si aggiunga questa linea e si rilanci il programma:

```
3Ø PRINT "ARRIVEDERCI"
```

"ARRIVEDERCI" apparirà subito sotto "CIAO". Supponiamo che si voglia spaziare "CIAO" da "ARRIVEDERCI".

Aggiungiamo questa linea e vediamo:

```
25 PRINT
```

Adesso ci sarà una linea vuota inserita fra "CIAO" e "ARRIVEDERCI".

Ora si ha un certo grado di controllo su come i messaggi vengono ordinati verticalmente sullo schermo, ma ancora nessun controllo su dove il messaggio compare su ogni linea. Supponiamo che si voglia scrivere "CIAO" verso la metà della linea. Il computer ha un comando chiamato PRINT TAB che permette di disporre il messaggio dovunque si voglia sulla linea. Quando si aggiunge questa linea e si dà il RUN, si vedrà l'effetto di PRINT TAB:

```
4Ø PRINT TAB(15);"BUONA SERA"
```

Gli spazi su ogni linea dello schermo sono numerati da Ø a 31 a partire da sinistra. Il messaggio alla linea 4Ø, quindi, comincerà dallo spazio numero 15. Attenzione a non lasciare spazi tra la TAB e la parentesi, altrimenti (15) verrà preso come variabile e TAB non funzionerà. Ci si assicuri anche di inserire il punto e virgola. TAB abbrevia le tabulazioni. Questo programma mostra quanto possa essere utile TAB:

```
1Ø CLS
2Ø PRINT"NUMERO";TAB(7);"AL CUBO";TAB
   (13);"ALLA 2A";TAB(18);"RADICE";TAB
   (26);"RECIPR."
3Ø FOR J=1 TO 12
4Ø PRINT TAB(1);J;TAB(6);J*J*J;TAB(12);J*J;
   TAB(17);INT(SQR(J)*1ØØØ)/1ØØØ;TAB(25);
   INT(1ØØØ/J)/1ØØØ
5Ø NEXT J
```

Dopo il RUN, si vedrà formarsi una tabella di numeri. Ogni volta che il programma entra nel ciclo, viene visualizzata un'altra linea di numeri sullo schermo. L'uso di PRINT TAB permette di sistemare ogni numero nella colonna giusta.

Se non si è afferrato il perché della moltiplicazione e della divisione per 1ØØØ alla linea 4Ø, la spiegazione è che questa riduce il numero dei decimali a tre. Supponiamo di voler stampare in un punto preciso dello schermo e non sulla prima linea disponibile. In questo caso, PRINT TAB non basta. Al suo posto si deve usare PRINT @. Si provi questo programma:

```
1Ø CLS
2Ø PRINT@ 463,"BASSO"
3Ø PRINT@ 71,"ALTO"
4Ø PRINT@ 257,"SINISTRA"
5Ø PRINT@ 282,"DESTRA"
```

Si vedranno comparire le parole sullo schermo nelle giuste posizioni.

I numeri dopo PRINT@ si riferiscono alle locazioni dello schermo, che su Dragon e Tandy sono numerate da sinistra a destra a partire dall'alto. Ecco un programma che usa PRINT @ e PRINT TAB:

```
10 CLS
20 PRINT "SQUADRA";TAB(7);"IN CASA";TAB
   (13);"FUORI";TAB(19);"PART.";TAB(26);
   "MEDIA"
30 PA=32
40 PRINT@448," ":PRINT@448,"SQUADRA
   □";:INPUTN$
50 PRINT@448," ":PRINT@448,"IN CASA□";:
   INPUT IN
60 PRINT@448," ":PRINT@448,"FUORI□";:
   INPUT NO
70 PRINT@448," ":PRINT@448,"PARTITE□";:
   INPUT RU
80 PRINT@PA,N$;TAB(7);IN;TAB(12);NO;TAB
   (18);RU;TAB(24);
90 IF RU=0 THEN PRINT"□
   *" ELSE PRINT INT((IN+NO)/RU*100)
   /100
100 PA=PA+32
110 IF PA<448 THEN GOTO 40 ELSE END
```

Quando si esegue il programma, la linea 20 stampa le intestazioni delle colonne in cima allo schermo. Le linee da 40 a 70 liberano una linea sullo schermo e chiedono informazioni per la INPUT. La linea 80 tabula le informazioni fornite e la 90 calcola la media. Se la media tende a infinito, nel caso la macchina provi a dividere per zero, è visualizzato un asterisco.

La variabile PA permette alla macchina di controllare che lo schermo non sia stato riempito. Quando la tabella ha riempito lo schermo, il programma si ferma.

## PUNTEGGIATURA

La punteggiatura è molto importante nel visualizzare informazioni su schermo. Virgole e punti e virgola controllano la posizione del cursore. Trascriviamo:

```
10 CLS
20 PRINT "IL CURSORE TORNA A CAPO"
30 PRINT "SENZA IL PUNTO E VIRGOLA"
40 PRINT "MA COL PUNTO E VIRGOLA□";
50 PRINT "ACCADE QUESTO!"
```

Quando si inserisce un punto e virgola in una linea di programma (fuori dalle virgolette), il cursore non va all'inizio della linea seguente, ma rimane invece esattamente dov'era la termine della visualizzazione. La successiva istruzione PRINT continua subito dopo l'ultima. Si aggiungano queste linee al programma e si dia il RUN:

```
60 PRINT "E SE NOI",
70 PRINT "USASSIMO",
80 PRINT "ALCUNE",
90 PRINT "VIRGOLE?"
```



Lo schermo del computer viene diviso in due metà. Quando si mette una virgola al termine di una linea di PRINT, il cursore salta nella metà accanto dello schermo. Le virgole possono quindi fornire una rudimentale forma di tabulazione.

Controlli e cursore, punteggiatura, funzione POKE e PRINT e spaziatura si possono tutti usare (singolarmente o assieme), per dare ordine alla scrittura del Commodore sullo schermo.

## FUNZIONI DI PRINT

La funzione più comunemente usata per il posizionamento del cursore è forse la TAB. Questa si comporta come il tabulatore di una macchina da scrivere, posizionando il cursore di un certo numero di spazi carattere, o colonne, dal margine sinistro dell'area del testo. Prende la forma:

```
10 PRINT TAB(10) "CIAO"
```

Dopo le parentesi, si può mettere un punto e virgola, ma non è necessario. *Non* ci deve essere spazio tra TAB e l'*argomento* tra parentesi, il cui valore è il punto di partenza della stringa stampata. Si possono usare ripetute TAB per posizionare diversi testi su ogni linea dello schermo.

```
10 A$="BUON": B$="GIORNO"
20 PRINT TAB(5)A$ TAB(10)B$
```

Il risultato è che B viene visualizzato alla colonna 5 e G alla colonna 10: il valore dell'argomento inizia sempre dal margine sinistro dell'area del testo.

Ora cambiamo la linea 20 come segue:

```
20 PRINT TAB(5)A$ SPC(5)B$
```

Ancora uno schermo dello Spectrum; uso delle virgole, che incolonnano i dati, con un programma come:

```
10 PRINT "messaggio",
20 GOTO 10
```

I computer Acorn e Commodore dispongono di più colonne.

Notiamo che ora la funzione SPC crea 5 spazi fra le due parole, partendo dalla fine della parola precedente. Cambiamo l'argomento alla 115: notiamo come la spaziatura continui per diverse linee. Non si possono usare valori oltre 255. Si possono includere diverse funzioni SPC nella stessa istruzione PRINT, ma si ricordi che lo schermo è limitato a 25 linee di 40 colonne e il totale degli spazi carattere non può superare questo limite. TAB e SPC possono usare variabili nel loro argomento, come in questo programma. Sul Vic 20, si cambi, alle linee 10 e 40, il 10 in 40.

```
5 PRINT"▢"
10 FORL=1 TO 10
20 PRINT TAB(L)" ▢ "SPC(L+1)"."SPC
   (L+2)" ▢ "
30 NEXT:R=L
40 FORL=10 TO 1 STEP-1
50 PRINT TAB(L)" ▢ " SPC(R+1)"."SPC
   (R+2)" ▢ "
60 NEXT
```

Questo visualizza una semplice prospettiva di una strada. È un esempio di uso inconsueto per funzioni normalmente impiegate per le maschere di informazioni, tipo i menu. Spesso trascurata, ma pur sempre appartenente allo stesso gruppo, è la funzione POS, che fornisce il valore della colonna (da 0 a 39) dove inizia la *successiva* posizione della PRINT. Questo programma stampa una strada a caso. Sul VIC 20, si cambi, alla linea 10, il 20 in 6 e, alla linea 30, il 25 in 12:

```
10 PRINT "▢": LET A=1: LET Z=20
20 LET Z=Z+A: PRINT TAB(Z)" ↑ □□□
   ↑ ";
30 IF POS(0)>25+RND(1)*9 OR POS(0)<5
   +RND(1)*9 THEN A=-A
40 PRINT:GOTO 20
```

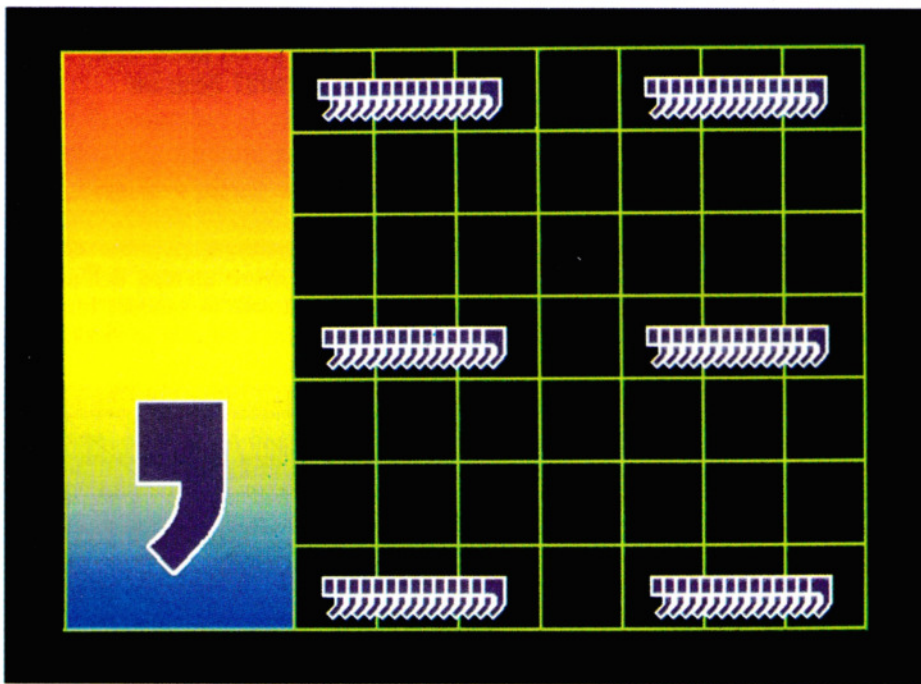
Il valore dopo POS, l'argomento, in questo caso è soltanto un valore di comodo.

## USARE GLI SPAZI

Elementi per la formattazione dello schermo possono venir realizzati incorporando spazi all'interno di istruzioni PRINT e, per piccoli spazi, questo metodo occupa meno memoria.

```
10 PRINT "▢□□□EHI, CIAO!"
```

Comunque è del tutto corretto includere stringhe anche lunghe di spazi, quando queste vengano usate con scopi di formattazione, assieme alla "segmentazione delle stringhe" (esposta più avanti).

Uno dei vantaggi meno evidenti di usare gli spazi in questo modo è che essi sono effettivamente riprodotti come caratteri

**3. Le virgole in istruzioni PRINT o INPUT significano che le informazioni verranno riprodotte incolonnate. Lo Spectrum e il Dragon hanno due colonne di tabulazione, il Commodore ne ha quattro e i computer dell'Acorn hanno una sceltra tra due, quattro o otto**

vuoti, sovrapponendosi così a qualsiasi maschera presente sullo schermo. TAB, SPC e caratteri di controllo del cursore non servono a questo, perciò talvolta è necessario includere spazi (e codici di "ritorno carrello" del cursore) insieme a queste funzioni, in modo da ottenere una vera e propria visualizzazione di spazi vuoti. Su questo torneremo più avanti.

## AIUTI PER IL POSIZIONAMENTO

Ogni volta che compare un'istruzione PRINT, visualizza qualcosa su una nuova linea. Così:

```
10 PRINT "■■■■■■■QUI SIAMO
   8 LINEE PIÙ IN BASSO RISPETTO ALLA
   PAROLA 'RUN'"
```

Questo sistema è semplice, ma consuma molta memoria: è meglio incorporare codici di controllo del cursore nella stringa da stampare. Dopo il NEW, scriviamo:

```
10 PRINT:PRINT:PRINT:PRINT:PRINT
20 PRINT "QUESTA SCRITTA È SETTE LINEE
   PIÙ IN BASSO DEL MESSAGGIO 'READY'"
```

Vari programmi in *Input* hanno fatto uso di questo e di altri controlli del cursore. Il problema di *cancellare*, accennato nel precedente paragrafo, può essere risolto con l'aggiunta di queste linee:

```
20 FOR D=1 TO 1000: NEXT
30 PRINT "■....E QUESTO CANCELLA LA
   SCRITTA □□□□□□□□□□□□□□□□
   □□□□"
```

Qui, il "cursore verso l'alto", incorporato all'inizio della linea 30, fa sì che la visualizzazione si sovrapponga a ciò che è già sullo schermo. Un uso più comune però è come parte di una routine di INPUT:

```
10 PRINT "■■■□□SQUADRA□□□□
   IN CASA□□□□FUORI□□□PART.□□
   MEDIA□□□□":OPEN1,0
20 PRINT "■"SPC(1);:INPUT#1,N$:PRINT
22 PRINT "■■"SPC(10);:INPUT#1,E:
   PRINT
24 PRINT "■"SPC(20);:INPUT#1,R:PRINT
26 PRINT "■"SPC(27);:INPUT#1,N:PRINT
28 IF N<>0 THEN PRINT"■■"SPC(31);
   INT((E+R)/N*100)/100:GOTO 40
30 PRINT "■■";SPC(32);"*"
40 GOTO 20
```

In questo programma sulla media di goal per partita, il cursore salta automaticamente da colonna a colonna, aspettando un INPUT, prima un nome (senza virgole!), poi cifre per numeri di goal "in casa" e goal "fuori casa", dopodiché visualizza la media calcolata.

I simboli alla linea 10 rapresentano CLR/HOME, bianco, RVS ON. Quello alla linea 20 sceglie il nero, quelli alla 22 il "cursore in basso" e il rosso, mentre il simbolo con le frecce all'esterno rappresenta la scelta del verde.

## PUNTEGGIATURA

Parleremo ora dell'uso della virgola o del punto e virgola relativi alla scrittura sullo schermo. Aggiungendo un punto e virgola al termine di un'istruzione PRINT, si può collegare una sequenza di altre PRINT:

```
10 FOR A=1 TO 400: PRINT "A";: NEXT
```

Il programma visualizza dieci linee piene di A. Può sembrare inutile, ma non lo è: si sostituisca un carattere spazio A dopo PRINT e si hanno le basi per una routine di pulizia dello schermo (sul VIC 20, cambiare i 400 in 220):

```
10 FOR A=1TO400:PRINT"A";:NEXT
15 FOR D=1 TO 1000:NEXT
20 PRINT"■■■■■■■■■■■
   ■"
30 FOR A=1TO400: PRINT"□";:NEXT
```

Questo dapprima stampa le dieci linee di A, poi vi sovrappone gli spazi bianchi.

Anche le virgole adempiono a una funzione: si può usare una sequenza di stringhe per formattare una specie di tabella. Sul Commodore 64 si digiti:

```
10 PRINT"■COLONNA 1","COLONNA 2",
   "COLONNA 3","COLONNA 4"
20 PRINT "■■ELEM. 1","ELEM. 2"
30 PRINT"ELEM. 3","ELEM. 4"
```

La linea 10 separa i titoli usando virgole. Si osserva che ogni elemento inizia alla colonna 0, 10, 20 o 30. Queste posizioni delle colonne sono fisse: le linee 20 e 30 mostrano come si possono usare virgole per saltare da una all'altra aggiungendo o togliendo altre virgole.

## POSIZIONAMENTO DEL CURSORE

I codici di controllo del cursore si possono usare per riprodurre le facilitazioni offerte della PRINT AT di altre macchine.

Si parta col definire due stringhe di controlli cursore, la prima di '39 cursore a destra' (21 sul VIC 20), la seconda di '24 (22) cursore in basso'.

```
10 C$="■■■■■■■■■■■■■
   ■■■■■■■■■■■■■
   ■■■■■■■■■■■■■
   ■■"R$="■■■■■■■■
   ■■■■■■■■■■■■■
   ■■■"
```

Ora, per disporre nel punto voluto la visualizzazione della PRINT, basta scegliere la posizione di X (colonna) e Y (riga) all'interno della linea del programma seguente. Una volta definita, la stringa di controlli cursore si può usare in altre linee:

```
500 PRINT "■"; LEFT$(C$,X);LEFT$(R$,Y);
    "COMMENTO"
```

# RIPRODURRE LETTERE CON FACILITÀ

**Come evitare errori nella stesura di lettere importanti? Ecco un programma per la redazione di un originale e per produrre poi le copie da mandare a persone diverse.**

Scrivere una grande quantità di lettere simili, per esempio circolari, può essere un lavoro lungo e noioso, se fatto a mano. Il programma presentato in queste pagine si propone di eliminare la parte ingrata di tale compito. È una versione semplificata di un "text editor", che permette di immettere un originale e poi produrne copie a volontà, ciascuna leggermente diversa dall'altra se necessario. Alcune tastiere semplificano il lavoro più di altre: non viene data una versione per lo ZX81, per la limitazione della sua tastiera. Sono disponibili come accessori per lo Spectrum tasterie 'maggiorate'.

Diversamente da un vero e proprio programma di *word-processing*, che richiederebbe ore per essere digitato, non è possibile (eccetto che sulle macchine Acorn) vedere esattamente sullo schermo TV l'aspesso finale della lettera. Ciò non è un problema: è lo stesso programma ad evitare, automaticamente, di "spaccare" le parole alla fine delle righe. Inoltre, viene anche automaticamente inserita una riga di spazio fra capoversi: più ordinato e professionale del "rientro". Per lettere importanti, è ovviamente necessaria una buona stampante (e un buon nastro!). È costosa, certo, ma se non se ne possiede una, può darsi che qualche amico sia disposto a farcene un prestito. I vari tipi di stampante verranno esaminati in una successiva lezione.

## IMMISSIONE DEL PROGRAMMA

Si inizia digitando il programma principale, senza assegnazioni di DATA. Poi lo si salva su nastro o dischetto, per poterlo usare, in seguito, in più occasioni. Prima di stampare la prima lettera, può esser necessario qualche adattamento, in base alla stampante che si intende usare.

Prima di tutto, si verifichino quanti caratteri vengono accettati dalla stampante su una linea. Quindi, si adatti il programma alla lunghezza stabilita della linea, e allo spessore del margine scelto sui due lati. In tutti i programmi, TL rappresenta la larghezza totale disponibile e LL la larghezza del testo della lettera. Entrambe possono naturalmente essere cambiate per adattarle, se necessario, a esigenze particolari.

E proprio in questo aggiustamento sta la differenza fra un computer e l'altro.

Si corregga la linea 3Ø in modo che TL sia la larghezza disponibile della stampante e LL la larghezza voluta per la lettera.

Volendo visualizzare la lettera sullo schermo, per avere un'idea dell'aspetto finale della lettera, si cancelli temporaneamente la linea 7Ø. Ma la si ripristini per la stampante.

Si corregga la linea 1Ø, in modo che PL sia la lunghezza disponibile della stampante e LL la lunghezza voluta per la lettera. (Nell'esempio offerto, le larghezze sono quelle disponibili sulla stampante ZX dello Spectrum, anche se la qualità di quest'ultima non è l'ideale dovendo scrivere una lettera importante).

Si corregga la linea 2Ø in modo che TL sia la larghezza disponibile della stampante e la 3Ø in modo che LL sia la larghezza voluta per la lettera.

Così com'è, la linea 4Ø permette di vedere sullo schermo dove le parole *vanno a capo*. Prima di stampare la lettera, si cambi P=Ø in P=2.

Si corregga la linea 1Ø in modo che TL sia la larghezza disponibile della stampante e che LL sia la larghezza desiderata per la lettera.

N.B. Il programma del Commodore è stato provato soltanto su stampante Commodore.

## IMMISSIONE DELLA PRIMA LETTERA

La lettera consiste in una serie di istruzioni DATA. Si inseriscano come mostrato nell'esempio a pagina 128 (è per il Dragon, ma le immissioni che servono vanno bene per tutti i computer).

Le DATA devono partire alla linea 1000 e devono cominciare con l'indirizzo del mittente.

La prima cosa che il computer farà sarà di scorrere le linee che contengono questo indirizzo, trovare la linea più lunga e disporre ordinatamente l'intero indirizzo lungo il lato destro del foglio.

Dopo di ciò, il computer farà scorrere ogni inizio di linea a sinistra, secondo uno stile adottato nelle lettere d'affari.

Va sottolineato che ogni linea della lettera deve iniziare con le virgolette, come avviene per la maggior parte delle assegnazioni di DATA.

Ecco un elenco dei significati degli altri simboli:

- Il segno # significa "questa linea fa parte dell'indirizzo del mittente: disporre a destra della pagina".
- Il segno $, significa "inizio di un capoverso", lasciare sopra di esso una riga vuota".
- La e commerciale, &, significa "inizio di nuova linea, scorrere a sinistra senza lasciare spazio sopra di essa". (Questo è utile per l'indirizzo del destinatario).
- Il segno moltiplicatore *, significa "centra questa linea (soltanto)".

Se si vogliono ulteriori linee di spaziature (per esempio, tra "Cordiali saluti" e la firma), non c'è da fare altro che inserire qualche segno di dollaro in più, ciascuno racchiuso tra virgolette.

Alla fine della lettera, due dei computer richiedono una linea supplementare:

Inserire un altro numero di linea seguito da DATA$.

Inserire un altro numero di linea, seguito da DATA.

## CORREGGERE LA LETTERA

Dal momento che è tutto in BASIC, è possibile, volendo, salvare la prima lettera (o bozza) come parte del programma originale. In questo caso, il primo passo nella correzione è caricare il programma più le DATA.

Poi, si può cambiare ogni linea semplicemente con i comuni interventi di *edit*.

Se si dimentica di salvare la prima lettera, occorrerà digitare nuovamente tutte le DATA, anziché apportare solo qualche correzione.

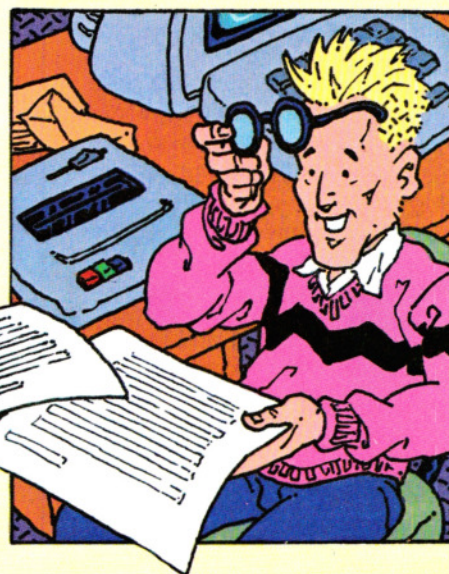
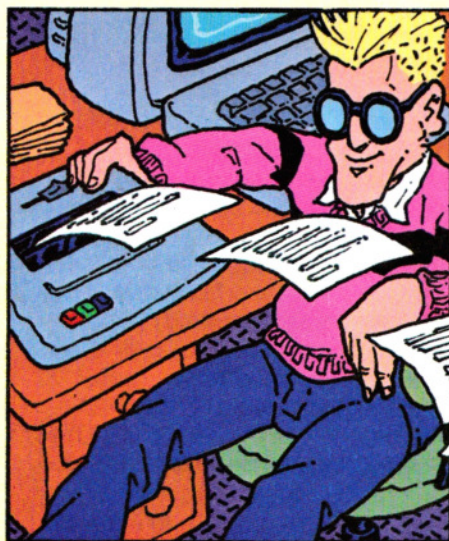
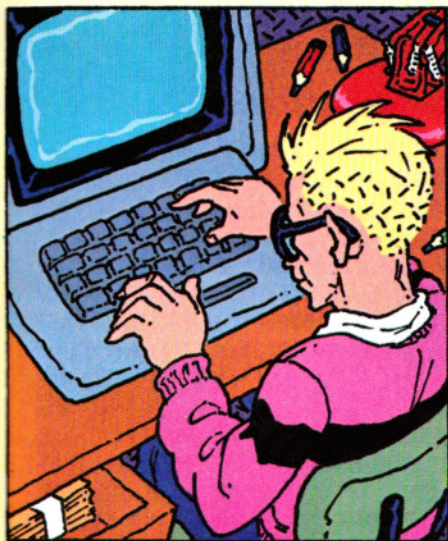
## STAMPA DELLA LETTERA

Per stampare la lettera basta semplicemente dare un RUN al programma, che attiva automaticamente la stampante e la disattiva dopo la stampa.

Se si lancia il programma senza aver collegato la stampante e senza ricordarsi di cancellare la linea 70, l'inizio della lettera comparirà sullo schermo, quindi il computer aspetterà che essa venga stampata. Si recupera la situazione interrompendo il programma, cancellando la linea 70 e ridando il RUN.

Stavola, si potrà vedere l'intera lettera sullo schermo.

Il tipo di carta su cui stampare dipende dal tipo di stampante usata. Alcune offrono una scelta di fogli singoli, migliori per una corrispondenza "ufficiale", altre di moduli continui.

Se si usano questi ultimi, si faccia scorrere la carta in modo che la lettera cominci su una pagina intera.

Dopo aver strappato i margini lungo le perforazioni, la pagina dovrebbe essere presentabile, ma, qualora si rendesse necessario, la si potrebbe rifilare con una taglierina.

Nel caso si possieda carta intestata, allora non si vorrà certo ristampare il proprio indirizzo.

Le DATA si fanno iniziare alla linea 1000, ma in questo caso la stampa comincia con l'indirizzo del destinatario e non servirà usare il segno #.

```
10 ON ERROR GOTO 480
20 MODE0
30 LL=60:TL=80
40 C$=""
50 LL=LL+1
60 D$=STRING$((TL-LL)/2,"□")
70 VDU2
80 PRINTD$;
90 READ A$
100 A$=A$+"□"
110 IF ASC(A$)=35 THEN PROCADDR
120 IF ASC(A$)=36 THEN PRINT"D$;:A$=
    RIGHT$(A$,LEN(A$)-1):LP=0
130 IF ASC(A$)=38 THEN PRINT'D$;:A$=
    RIGHT$(A$,LEN(A$)-1):LP=0
140 IF ASC(A$)=42 THEN PROCCENT:GOTO
    90
150 C$=""
160 FOR T=1 TO LEN(A$)
170 B$=MID$(A$,T,1)
180 C$=C$+B$
190 IF LEN(C$)>LL THEN VDU3:
    PRINT'"ERRORE DI FORMATO: PAROLA
    TROPPO LUNGA!":END
200 LP=LP+1
210 IF LP<=LL THEN 240
220 PRINT'D$;:IF ASC(C$)=32 THEN C$=
    RIGHT$(C$,LEN(C$)-1)
230 LP=LEN(C$)
240 IF B$="□" AND LP<=LL THEN
    PRINTC$;:C$=""
250 NEXT
260 GOTO 90
270 DEF PROCCENT
280 A$=RIGHT$(A$,LEN(A$)+-1)
290 IF LEN(A$)>LL THEN VDU3:
    PRINT'"LINEA TROPPO LUNGA: NON
    POSSO CENTRARLA":END
300 PRINT;'D$;STRING$((LL-LEN(A$))/2,
    "□");A$;'D$;
310 LP=0
320 ENDPROC
330 DEF PROCADDR
340 X=LEN(A$)
350 RESTORE
360 READ A$:IF ASC(A$)<>35 THEN 390
370 IF LEN(A$)>X THEN X=LEN(A$)
380 GOTO 360
390 IF LEN(A$)>LL THEN VDU3:PRINT'
    "ERRORE DI FORMATO: INDIRIZZO
    TROPPO LUNGO":END
400 E$=STRING$((LL-X),"□")
410 RESTORE
420 READ A$
430 REPEAT
440 A$=RIGHT$(A$,LEN(A$)-1)
450 PRINTE$A$'D$;
460 READ A$
470 UNTIL ASC(A$)<>35
475 ENDPROC
480 PRINT:VDU3:MODE6:REPORT:PRINT:END
```

```
10 LET LL=32: LET PL=32
15 LET LL=LL+1: LET T=(PL-LL)/2
20 LET D=0
30 READ A$: LET L=LEN A$
40 LET C=0
50 IF C=L THEN GOTO 30
60 LET C=C+1: LET D=D+1: IF C>1
   THEN GOTO 100
70 IF A$(C)="#" THEN GOTO 500
80 IF A$(C)="*" THEN GOTO 700
85 IF A$(C)="&" THEN GOTO 850
90 IF A$(C)="$" THEN LPRINT CHR$
   13;CHR$ 13;: LET D=0: GOTO 900
95 LET A$="□"+A$: LET L=L+1
100 IF A$(C)="□" THEN GOTO 800
110 LPRINT A$(C);
115 IF D>LL THEN LET D=0
```

```
120 GOTO 50
500 LET NL=0: LET TA=LL: LET BE=0
510 LET LE=LEN A$-1: IF LE>LL THEN
    PRINT FLASH 1;"ERRORE DI FORMATO:
    INDIRIZZO TROPPO LUNGO":STOP
520 IF LE>BE THEN LET BE=LE
530 LET NL=NL+1: READ A$: IF
    A$(1)="#" THEN GO TO 510
540 RESTORE 1000
550 LET TR=T+LL-BE: FOR G=1 TO NL:
    FOR H=1 TO TR: LPRINT"□";: NEXT H:
    READ A$: LPRINT A$(2 TO): NEXT G
560 GOTO 30
700 LET TA=(LL-L)/2+T: IF TA<T
    THEN LPRINT CHR$ 13: PRINT FLASH
    1;"ERRORE DI FORMATO: NON RIESCO A
    CENTRARE":STOP
710 LPRINT CHR$ 13;: FOR N=1 TO TA:
    LPRINT "□";: NEXT N: LPRINT A$(2 TO
    L): GOTO 20
800 LET SL=LL-D-1: LET CC=C+
    1: LET X=1
810 IF CC=L THEN GOTO 825
820 IF A$(CC)<>"□" THEN LET CC=CC
    +1: LET X=X+1: GOTO 810
825 IF X>=LL THEN PRINT CHR$ 13:
    PRINT FLASH 1;"ERRORE DI FORMATO:
    PAROLA TROPPO LUNGA":STOP
830 IF SL>=X THEN GOTO 110
850 LPRINT CHR$ 13;: LET D=0
900 FOR B=1 TO T: LPRINT "□";: NEXT B:
    GOTO 50
```

```
10 CLEAR 200
20 TL=80
30 LL=56
40 P=0
50 SP=(TL-LL)/2:HW=TL/2
60 PRINT #-P,CHR$(13)
70 IFP=2 THEN SP$=STRING$(SP,"□")
   ELSE SP=0:HW=16
200 READA$:A$=A$+"□"
210 IF A$="" GOTO 290
220 O$=LEFT$(A$,1)
230 IF O$="#" THEN ML=0:GOSUB 400:
    GOTO 290
240 IF O$="$" THEN RL=0:PRINT#-P,
    CHR$(13);CHR$(13);SP$;:ST=2:GOTO 280
250 IF O$="*" GOSUB800:RL=HW-LEN
    (A$)/2:GOTO280
260 IF O$="&" THEN RL=0:PRINT#-P,
    CHR$(13);SP$;:ST=2:GOTO 280
270 ST=1
280 GOSUB 600
290 GOTO200
400 IF LEN(A$)>ML THEN ML=LEN(A$)
410 N=N+1:READA$:IF LEFT$(A$,1)=
    "#" GOTO 400
420 IF ML>LL THEN CLS:PRINT"errore di
    formato□...INDIRIZZO TROPPO LUNGO
    PER IL MARGINE ATTUALE!":END
430 RESTORE:FOR J=0TO N-1:READA$:
    PRINT#-P,STRING$(LL-ML,"□");SP$;:
    PRINT#-P,MID$(A$,2);CHR$(13);
440 NEXT:RETURN
600 WL=0
610 IF ST+WL>LEN(A$) THEN 630
620 IF MID$(A$,ST+WL,1)<
    >"□" THEN WL=WL+1:GOTO 610
630 IF WL>LL THEN CLS:PRINT"errore di
    formato...";A$:PRINT"□...CONTIENE UNA
    PAROLA PIÙ LUNGA DELLA INTERA
    LINEA!":END
640 IF RL+WL-1>LL THEN PRINT#-
    P,CHR$(13);SP$;:RL=0
650 WL=WL+1
660 PRINT#-P,MID$(A$,ST,WL);:RL=RL
    +LEN(MID$(A$,ST,WL))
670 ST=ST+WL:IF ST<LEN(A$)+1
    GOTO 600
680 RETURN
800 IF LEN(A$)>LL THEN CLS:PRINT
    "errore di formato...NON POSSO
```

**Quale dev'essere l'aspetto di una domanda di impiego?**

La stesura e la presentazione di una domanda di impiego è estremamente importante, dato che offre al potenziale datore di lavoro la prima impressione dello scrivente. Una lettera disordinata o frammentaria dà una cattiva impressione, indipendentemente dalle sue qualità intrinseche. Lo stile di redazione di una lettera cambia negli anni, ma queste sono le regole più aggiornate.

L'indirizzo del mittente va a destra in alto sul foglio, ma non troppo vicino all'angolo. Si possono tralasciare tutti i segni di interpunzione e si dovrebbe iniziare ogni linea subito sotto la precedente, senza intervalli.

La data va sotto l'indirizzo, dopo una linea di spazio se lo si desidera.

Poi viene il nome e l'indirizzo del destinatario. Questo va sul lato sinistro della pagina o al livello o sotto l'indirizzo del mittente. Nome e indirizzo dovrebbero venire scritti per intero, presumendo che si conservino copie o registrazioni. Se si sta rispondendo a un annuncio di lavoro, in cui è citato un codice di riferimento, questo andrebbe inserito sopra l'indirizzo.

Se si sta facendo una domanda per un incarico particolare, la si scriva al centro della linea sotto 'Egr. Signore', o che altro, e si lasci uno spazio sopra e sotto, in modo che chi seleziona la domanda possa capire con un'occhiata di cosa si tratta.

Il corpo principale della lettera ha un'apparenza più ordinata se i capoversi iniziano a sinistra con uno spazio tra l'uno e l'altro. Infine, 'In fede', o 'Cordiali saluti' possono andare a sinistra, a destra o al centro della pagina col nome dello scrivente sottolineato e abbastanza spazio per la firma.

**Un fac-simile di lettera (immessa come una serie di frasi DATA, nel Dragon), riprodotta dalla stampante**

```
    CENTRARE",A$:END
810 PRINT # −P,CHR$(13);
820 IF HW > LEN(A$)/2 THEN PRINT # −P,
    STRING$(HW − LEN(A$)/2,"□");
830 ST = 2:RETURN
```

```
10 CLR:LL = 60:TL = 80:QQ = (TL − LL)/2:
    DIM A$(20):PRINT"♥"
15 FOR Z = 1 TO LL + QQ:S$ = S$ + "□":
    NEXT Z
20 OPEN 4,4:CMD 4:LE = 0:GOTO 800
100 READ K$
102 K$ = K$ + "□":IF LEFT$(K$,1) = "*"
    THEN LE = 0:GOTO 600
110 IF K$ = "□" THEN PRINT # 4,CHR$(13):
    CLOSE 4:END
115 IF LEFT$(K$,1) = "&" THEN LE = 0:
    PRINT # 4,CHR$(13);:GOTO400
120 IF LEFT$(K$,1) = "$" THEN PRINT # 4,
    CHR$(13):LE = 0:GOTO 400
130 GOTO 452
400 K$ = RIGHT$(K$, LENK$) − 1)
450 IF LEN(K$) < 1 THEN 100
452 IF LE = 0 THEN PRINT # 4,LEFT$(S$,
    QQ);
455 FOR L = 1 TO LEN(K$)
460 IF MID$(K$,L,1) = "□" THEN KK$ =
    LEFT$(K$,L):K$ = RIGHT$(K$,LEN(K$) −
    L):GOTO 480
470 NEXT:GOTO 100
480 GOSUB 500:GOTO 450
500 IF LEN(KK$) > LL + 1 THEN 950
505 IF LE + LEN(KK$) > LL THEN PRINT # 4,
    CHR$(13); LEFT$(S$,QQ);:LE = 0
510 LE = LE + LEN(KK$):PRINT # 4,KK$;:
    RETURN
550 RETURN
600 PRINT  # 4, CHR$(13); LEFT$(S$),QQ);:IF
    LEN(K$) > LL THEN 960
610 PRINT  # 4, LEFT$(S$,((LL*.5) − (LEN
    (K$) − 1)*.5)) RIGHT$(K$,LEN(K$) − 1):
    GOTO 100
800 READ X$:P = P + 1:IF LEFT$(X$,1) < >
    "#" THEN 900
810 A$(P) = RIGHT$(X$, LEN(X$) − 1):IF LEN
    (A$(P)) > HL THEN HL = LEN(A$(P))
815 IF HL > LL THEN 950
820 GOTO 800
900 IF P = 1 THEN RESTORE:GOTO 100
910 P = P − 1:FOR Z = 1 TO P
920 PRINT # 4, LEFT$(S$,QQ);LEFT$(S$,(LL
    − HL))A$(Z):NEXT Z:K$ = X$:GOTO 102
950 PRINT # 4,"□":PRINT "■ERRORE DI
    FORMATO: PAROLA TROPPO
    LUNGA":CLOSE 4:END
960 PRINT # 4,CHR$(13):PRINT "■ERRORE
    DI FORMATO: NON POSSO CENTRARE LA
    LINEA":CLOSE 4:END
```

# INDICE CUMULATIVO DEI FASCICOLI

# NEL PROSSIMO NUMERO

☐ *I giochi "arcade" sono scarsamente avvincenti se manca un avversario in grado di rispondere al nostro "fuoco". Vediamo come creare NEMICI MORTALI e come proteggersi con schermi.*

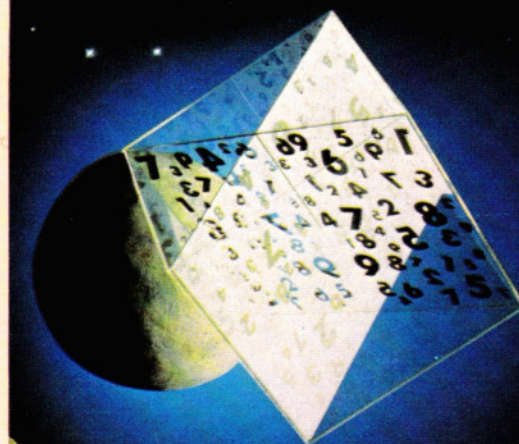
☐ *Mettiamo nuovamente al lavoro il computer con il nostro programma per elaborare il BILANCIO FAMILIARE.*

☐ *Un altro passo verso il codice macchina: impariamo la NUMERAZIONE ESADECIMALE, ossia il sistema a base 16 adoperato dal computer.*

☐ *Comunichiamo le nostre richieste al computer tramite i comandi INKEY$, GET e GET$ del BASIC e usiamo tali comandi in un programma di grafica.*

☐ *Scopriamo l'utilità delle MATRICI, uno strumento indispensabile offertoci dal BASIC per conservare le informazioni.*

INPUT 5

CORSO PRATICO DI PROGRAMMAZIONE PER LAVORARE E DIVERTIRSI COL COMPUTER

## CHIEDETE INPUT AL VOSTRO EDICOLANTE